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GUI测试脚本修复系统的设计与实现

作 者 姓 名 陈笑智

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

2019 年 5 月 27 日

学　　号：**MF1732016**

论文答辩日期：**2019 年 5 月 14 日**

指 导 教 师：　　（签字）

The Design and Implementation of Android GUI Test Script Repairing System

By

Xiaozhi Chen

Supervised by

Professor **Zhenyu Chen**

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

Software Institute

May 2019

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：安卓GUI测试脚本修复系统的设计与实现

工程硕士（软件工程领域） 专业 2017 级硕士生姓名：陈笑智

指导教师（姓名、职称）：陈振宇 教授

摘 要

随着移动互联网的不断发展，移动应用已成为人们工作生活中不可缺少的一部分，其中安卓应用占据了相当大的市场。安卓应用的快速迭代，使得全面的回归测试面临挑战。移动应用版本变更往往改变界面（GUI）结构和操作流，从而导致原有GUI测试脚本失效，无法直接复用。企业需要投入大量资源人工审查脚本代码，大大影响了应用的质量保障和发布周期。因此，提高测试脚本复用性对于移动应用质量保障具有重要价值。

本文设计与实现了一个安卓应用GUI测试脚本修复系统，通过修复因版本更新而失效的GUI测试脚本来提高复用性。系统将修复过程分为了四个部分，首先使用自动化遍历工具提取安卓应用的GUI组件，建立事件流图模型，图结构能够简洁直观地反映安卓应用的GUI特性。然后提取测试脚本代码中定位GUI组件的数据，为脚本代码与模型的建立映射关系。进而构建自动修复算法，基于事件流图和映射关系数据，采用弗洛伊德算法来对脚本进行初步修复。最后在自动化修复的基础上，提供人机协同修复的方式，进一步提升修复效果。在系统构建方面，本文采用面向服务架构，使用Spring Boot构建服务端程序，使用Dubbo实现修复服务的远程过程调用。系统使用Mybatis进行数据持久层管理，使用Angular来构建前端程序，系统前后端分离通过Http进行交互，使用Nginx进行反向代理和负载均衡。在移动应用测试框架方面，系统使用Appium和Uiautomator构建安卓自动化遍历工具和实现测试脚本的真机运行。

本文对安卓GUI测试脚本修复系统的可用性进行了初步实验评估。实现选取了不同版本的7款开源安卓应用和对应的GUI测试脚本进行修复实验，经过人工验证和统计，修复平均准确率达到80.27%。安卓GUI测试脚本修复系统能够提高安卓应用GUI测试脚本的复用性，提高测试人员维护脚本的效率，加快移动应用软件开发迭代流程。

关键词： 安卓GUI测试，测试脚本修复，事件模型，人机协同

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 Android GUI Test Script Repairing System

SPECIALIZATION: Software Engineering

POSTGRADUATE: Xiaozhi Chen

MENTOR: Professor Zhenyu Chen

Abstract

With the continuous development of mobile Internet, mobile applications have become a indispensable part of people's lives and Android applications occupy a considerable market. The rapid iteration of Android applications challenges comprehensive regression testing. Version changes often change the interface (GUI) structure and operation flow, which leads to the failure of the original GUI test script and can not be reused directly. Enterprises need to invest a lot of resources to manually review script code, this affects the quality assurance and release cycle of applications. Therefore, improving test script reusability is of great value to mobile application quality assurance.

This thesis designs and implements an Android application GUI test script repair system, which improves reusability by repairing GUI test scripts that have failed due to version updates. The system divides the repair process into four parts. Firstly, the GUI components of Android applications are extracted by using the automated traversal tool and the event flow graph model is established. The graph structure can concisely and intuitively reflect the GUI characteristics of Android applications. Then the data of GUI components' locator in the test script code is extracted and the mapping relationship between the script code and the model is established. Then an automatic repair algorithm is constructed based on event flow graph and mapping relationship data, Freud algorithm is used to preliminarily repair scripts. Finally, on the basis of automated repair, the method of man-machine collaborative repair is provided to further improve the repair effect. In the aspect of system construction, this thesis adopts service-oriented architecture and uses Spring Boot to build a server-side program and

Dubbo to implement remote procedure call for repair service. The system uses Mybatis to manage data persistence layer, Angular to build front-end program, Http to interact with front-end and back-end separately, Nginx for reverse proxy and load balance. In the aspect of mobile application testing framework, this thesis uses Appium and Uiautomator to build Android automated test input generation and execution tool and implement the real-time running of test scripts.

This thesis preliminarily evaluates the availability of Android GUI test script repair system. Seven different versions of open source Android applications and corresponding GUI test scripts are selected for repair experiments. After manual verification and statistics, the average accuracy of repair reaches 80.27%. Android GUI test script repair system can improve the reusability of test script in Android application GUI regression test, improves the efficiency of testers maintaining scripts, and accelerate the iteration of mobile application software development.

Keywords: Android GUI Test, Test Script Repair, Event Model, Human-machine Collaboration

目　录

表　目　录

图　目　录

第一章　引言

1.1　选题背景及意义

随着移动互联网技术的不断发展，移动设备已经成人类生活、信息传递的最流行的媒介，这就导致了移动互联网用户的爆发式增长。据有关机构统计，截止2018年6月，中国移动互联网用户数量已经达到了11亿。越来越多的行业都将自己的业务拓展到移动应用市场上，从而诞生了各种各样的移动应用。在众多的移动设备系统中，安卓占据了相当大的市场份额。为了保障安卓应用的质量，自动化测试技术成为安卓应用开发行业的重要模块，最基本就是GUI（Graphic User Interface，图形化用户界面）测试 [1]。随着应用的快速迭代，回归测试成为了非常重要的环节，而版本的变更很多时候会带来应用的界面结构和操作流的变化，从而导致很多旧版本的GUI测试脚本失效 [2]。这些测试脚本包含了测试人员的领域知识和设计思想 [3]，为了保障回归测试工作，测试人员不得不修复这些脚本。修复的方式主要是测试人员重新审查测试脚本代码，手工逐个地改编码来使这些失效的脚本重新变得有效 [4]，这种方式消耗了测试人员大量的精力和时间，造成了大量测试资源的损耗。

从2003年开始，学术界出现了很多关于软件测试脚本修复的研究，尤其在桌面应用、Web应用等方面涌现了不少关于GUI测试脚本修复的论文。但已有的研究产出大多停留在模型、算法层面，并未产出可用性较高的系统或者工具。而工业界已有的测试平台，如百度MTC[1]、阿里MQC[2]、腾讯WeTest[3]等，都不具备关于测试脚本修复的功能。

本文致力于实现一个基于事件流模型的安卓应用GUI测试脚本自动修复技术。事件流模型抽象了安卓应用的GUI特性，为GUI测试脚本修复提供了便捷的途径。当新版本的应用开发完成时，我们为新版本的应用生成事件流模型以建立测试脚本代码与模型的映射关系 [5]，并基于此映射关系来修复由于版本更新而失效的脚本。而修复效果很大程度上取决于所建立模型的准确度 [6]，而自动化技术所建立的事件流模型并不能达到百分之百的准确。为了解决这个问题，本文在自动化修复的基础上，进一步引入人机协同修复技术，完善脚本修复的效果 [7]。本系统可以提高测试脚本的复用性，减少移动应用版本更替时GUI测

[1] http://mtc.baidu.com/
[2] http://mqc.aliyun.com/
[3] http://wetest.qq.com/

试脚本修复的成本，提高测试人员对GUI脚本的维护效率，推动移动应用开发迭代的生产流程。

1.2　国内外研究现状

1.2.1　学术界现状

Memon最早关注到回归测试中由于版本变化带来的GUI测试用例不可复用的问题 [8]。他提出对于原始版本的GUI 测试脚本我们需要识别出其中可复用和不可复用的部分，并找出那些可以被修复的脚本并修复它们，以保证测试脚本的事件覆盖率并改善回归测试维护的效率。他使用事件流图（EFG）的方式来找出不同版本应用之间的事件区别，以确定那些可以被修复的测试脚本，事件流图上相邻节点的关系表示事件发生的前后顺序。在修复过程中，对于那些因为版本变化而变得非法的事件序列，Memon通过在序列中增加新的单个序列或者跳过那些非法的事件来获得一条完整事件序列，并依此将修复过程分为了四种类别。Memon所研究的GUI测试主要基于GUITAR框架，研究对象为桌面应用，所产出的工具和算法也仅仅是针对桌面应用 [9]。

Gao等提出了一种基于事件流图模型来进行GUI测试脚本修复的工具——SITAR。他们使用逆向工程工具GUI Ripper为应用近似的建立事件流图，为每一个测试用例建立与EFG的映射关系。他们使用脚本修复转换和人工输入来修复该脚本，并合成一个“已修复”的测试脚本。他们首次引入了dominates边（强制发生的前。事件），从顶点x到顶点y的dominates边表示事件y必须在事件x之前。由于自动逆向工程技术可能会错过应用的部分GUI组件，他们提出了一种人工辅助修复的方法，在修复过程中，由测试人员手动将有些边更新为dominates边。而SITAR不会单方面做出修复决定，所有修复均由测试人员授权，以确保修改不会导致测试脚本偏离测试的业务逻辑 [10]。同样SITAR 针对的仅仅是桌面应用，且没有产出可用性较高的工具或者系统。

Li等提出了一种基于事件序列模型（ESM）和增量事件序列模型（DESM）的建模方法来找到移动应用GUI中事件元素的关系序列，依此建立测试脚本与事件的映射。他们首先使用ESM来描述一个应用中的行为，ESM反映的主要是应用的各种状态和GUI元素。当一个新版本的应用出现时，他们通过向DESM中更新对应的事件变化，并基于这些更新来合成测试脚本以达到维护的目的。这样做的好处是建立模型的成本只在于一个新应用的第一次建模，之后就可以通过和测试脚本一起不断地以增量的方式来维护。维护的过程分为两个部分，第一部分对比测试脚本在ESM 上的模式和DESM发生的改变，从而合成出一个全

新的合法的模式。第二部分根据GUI事件与测试脚本映射关系将新的模式映射到测试脚本中 [11]。

Behrang等在提出了一种用于MOOCs移动应用开发课程作业评估中，使用同样一套测试脚本来测试不同学生开发的具备相同需求的应用。他们称之为测试迁徙，与本文要研究的场景有一定的相似之处。他们所提出的方法是通过切分，根据测试代码中的断言将代码分为多个片段，每个片段对应应用中的一个事件，然后为目标应用建立一个事件序列，他们通过匹配代码片段与事件序列中的事件来重新组装代码已达到测试脚本复用的目的 [12]。这种方式存在几点问题：1）现在比较流行的一些安卓测试脚本框架如Appium本身是没有断言的，Behrang 等人研究的方法局限于在使用Espresso作为测试框架的场景，2）测试代码重组的方式无法应对事件增加或减少的场景，这源于学生作业与应用迭代的本质区别。

从上述研究现状可以看出，现有的研究团队大多忽略了适量的人工干预对于脚本修复效果的作用，而且产出的工业级的工具或者平台很少，大多还停留在算法和实验层面。

1.2.2　工业界现状

目前国内的移动应用测试平台有腾讯旗下的WeTest[4]、百度旗下的MTC[5]、Testin云测[6]等，国外的测试平台有Test Cloud[7]、Test Droid[8]等。现有的移动应用测试平台在自动化测试方面大多提供以下两种服务：

1）提供测试脚本编写服务，即用户上传应用并支付一定的费用，由平台负责雇佣测试专家人工完成测试脚本的编写，用户只获得最终的测试报告。这种方式对于频繁迭代的移动应用来说，成本过高。

2）提供在线脚本录制和回放服务，即用户可在平台提供的虚拟界面执行GUI操作，平台通过云设备运行，并录制脚本。这种方式适用于大量GUI变动甚至是重构的情况，效果很好。但是对于GUI变动不是很大的版本变更，重新录制脚本是非常耗时的工作。

[4]http://wetest.qq.com/
[5]http://mtc.baidu.com/
[6]https://www.testin.cn/
[7]http://www.test-cloud.com/
[8]https://cloud.testdroid.com/

1.3　本文主要研究工作

本文所设计和实现的是安卓GUI测试脚本修复系统，是移动应用测试平台的一个子系统，主要功能是测试脚本的版本管理和脚本修复。本文首先分析了国内外现有研究成果的不足，对GUI测试脚本修复的具体方法和过程做了详细的分析，并对修复过程面临的诸多问题提出了对应的解决方案，对修复系统的设计和实现做了具体的描述。

系统对于测试脚本的修复工作主要借助于事件流图模型。安卓应用的GUI测试脚本的本质是模拟用户能够执行的操作事件从而对应用各项功能的正确性进行验证，而基于安卓应用GUI的特性可以分析出，安卓应用的GUI事件流可以抽象为图的结构 [13]。从而我们可以把由版本变更所带来的GUI 变化抽象为图模型的结构变化，如：新的GUI事件的加入在事件流图中体现为图节点的增加等。并且我们可以基于图的结构来构建一套自动化修复算法，针对于不同类型的GUI变化制定不同的修复策略。在修复过程中可以借助于图论的各种修复算法，如弗洛伊德最短路径算法等。

自动化修复的效果很大程度上取决于事件模型的准确度，现有的安卓自动化遍历算法的设计没法涵盖所有的事件流路径，在图模型上显示为边不够全面。从GUI测试脚本本身来讲，脚本中包含了测试人员对于测试策略的设计，这是自动化遍历工具所无法完全覆盖的。因此本文提出一种新的思路，引入人为干预的方式，通过人机协同修复的方式来提升脚本修复的准确度。

对于修复的效果，我们采用两种评估方式从两个方面进行评估，分别是代码层面和界面层面。总结为两个指标：SCP（Screen Coverage Presentation）和TAP（Test Action Presentation）。SCP即旧版本脚本覆盖的界面或事件维护后依然能被覆盖的比例。TAP即废弃的测试代码语句恢复可用的比例。两个指标越高，说明修复效果越好 [11]。

本文的系统采用前后端分离的方式开发，前端使用基于TypeScript语言的Angular框架进行开发，后端使用基于Java语言的SpringBoot框架进行开发，前后端使用Restful风格的Http方式交互，开发和调试效率非常高，服务调用通过Dubbo框架的RPC方式，轻量且高效。用户上传应用和脚本到系统，服务端连接手机，调用自动化遍历工具对应用进行扫描，并将模型反馈给用户，用户可以对模型进行修改，服务端实时调用手机服务，运行修改后的脚本，并将运行结果即脚本成功率计算后反馈给用户，如此来借助人机协同的手段完善脚本修复的效果。

1.4　本文的组织结构

本文描述了安卓GUI测试脚本修复系统的设计与实现过程，本文的组织结构如下：

第一章　对选题的背景和研究的目的和意义，国内外关于脚本修复的研究现状，以及相关脚本修复的工作内容了一个简单的介绍。

第二章　对本文中涉及和项目开发中所使用的相关技术做了详细介绍，包括Appium、Uiautomater、Spring Boot、Mybatis、Angular、D3.js 以及一些其他工具和技术等。

第三章　对GUI测试脚本修复系统的需求分析、架构设计、模块设计、数据库设计做了详细的分析和完整的描述，对模型生成等模块的业务交互流程做了初步设计。

第四章　对GUI测试脚本修复系统的模型生成模块、映射生成模块、脚本修复模块、人机协同模块四个模块的具体功能、详细设计和实现方案做了明确的分析和描述，并对每个模块所涉及的工作进行效果评估。重点阐述了脚本修复过程的实现。

第五章　对系统测试和实验设计所使用的安卓应用、测试脚本和过程的设计以及系统测试和实验的结果做了详细介绍。

第六章　对论文期间所做工作进行了总结，分析了下一步的工作，对安卓GUI测试脚本修复做了进一步展望。

第二章　相关技术综述

2.1　移动应用测试相关技术

2.1.1　Appium

Appium是一个开源的基于Nodejs编写的移动应用测试工具，支持Android和iOS平台上的移动应用。Appium具有跨平台的特性，能够允许测试人员使用相同的接口在不同的平台上编写自动化测试代码，显著提高了测试套件代码的复用性 [14]。

Appium是经典的C/S架构，核心是一个Web服务器，对外提供一套REST接口，用于接受客户端的连接、监听等各种命令，然后在移动设备上执行这些命令，并将执行结果以Http的方式返回给客户端。

Appium支持多语言，并针对主流的几种语言如Java、Python、JavaScript、Ruby、PHP等开发了对应的库，这些库实现了对WebDriver协议的扩展，提供了非常丰富的自动化API，如安装/卸载应用、各种事件操作等。Appium库封装了标准的Selenium客户端类库，以JSON格式为用户提供了所有常见的Selenium命令以及额外的移动设备控制相关的命令，如多点触控手势和屏幕朝向 [15]。

Appium完整的运行过程如图 2.1所示：Appium服务端调用Android adb工具完成基本的系统的操作，然后服务端向移动设备上部署一个Bootstrap.jar 中间件，之后Bootstrap.jar返回端口给服务端。服务端开启端口监听，默认情况下监听4723端口来接受客户端的请求，4724端口来与安卓设备通讯。服务端使用Webdriver协议分析命令并通过4724端口将分析后的命令发送给Bootstrap.jar，Bootstrap.jar接受到命令后转发给Uiautomator由其执行，关于Uiautomator的详情本文会在下一小节进行描述。Appium客户端启动时会初始化一个与服务端交互的Session，所有的自动化命令皆是在同一个Session上下文中运行的。首先，客户端会通过POST请求发送一个键值对JSON对象参数给服务端，服务端收到请求后会开启一个Session并将SessionID返回给客户端，客户端后续的命令都使用这个ID [16]。

本文研究的测试脚本为基于Appium编写的移动应用GUI测试脚本，这里主要考虑到其开源、跨平台等特性，在工业界具有广泛的应用，选取基于Appium编写的脚本作为研究目标更具有代表性和真实性。本文所使用的自研自动化遍历工具也是基于Appium来开发的。

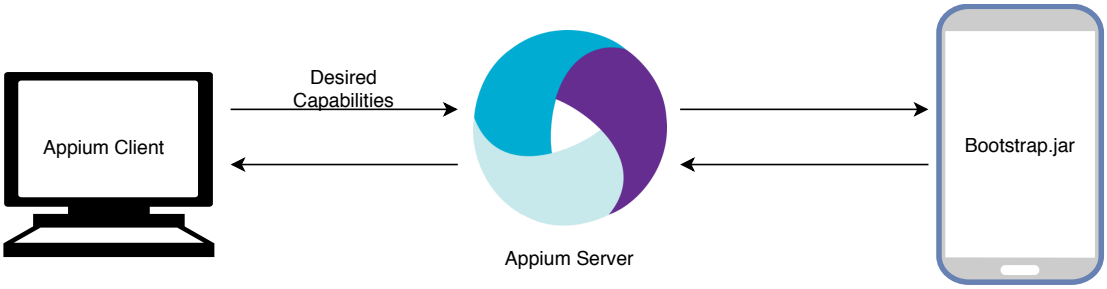

图 2.1: Appium运行原理图

2.1.2　Uiautomator

Uiautomator是安卓开发团队在Android4.1版本（API 16）中推出的一款UI自动化测试工具，一方面帮助开发人员更有效率地完成安卓应用的调试工作，另一方面也为测试人员提供了另一种更方便的自动化测试工具 [17]。

Uiautomator提供了两个工具来支持自动化测试。第一个是Uiautoviewer，这是一个用于分析应用图形界面上UI控件的工具，能够在应用运行时通过一定的指令获取当面界面上所有布局信息和空间元素数据的快照，如控件类型（class）、唯一标识（resouce-id）、控件文本信息（text）等。本文使用的自动化遍历工具在遍历过程中基于Uiautoviewer来实时获取界面数据以更新遍历控件的集合。第二个是Uiautomator库，这是一个基于Java语言的库，为自动化测试提供了很多丰富的API和执行自动化测试的引擎，如点击、滑动、键盘输入等 [18]，本文所使用的自研自动化遍历工具正是基于此特性来进行开发的。

Uiautomator具备以下几个优点。首先，它是Google安卓团队推出的，稳定性更好，维护和更新有很好的保证，而且能够获取更多的移动设备Android操作系统的权限。其次，它能够跨进程操作，这是其他其他自动化工具如Robotium、Instrumentation无法直接做到的。同时它的运行速度非常快，运行步骤也很简单。Uiautomator对于应用控件元素的定位方式很丰富，不仅支持使用控件本身的属性来定位，也支持使用坐标定位的方式。

本文采用Uiautomator进行安卓应用的GUI控件捕捉，从而构建自动化遍历工具，使用Uiautomatorviewer来进行事件流模型准确度的评估工作。

2.2　服务端相关技术

2.2.1　Spring Boot

Spring Boot 是Pivotal团队于2016年发布的一款全新的Java语言后端开发框架。其设计目的是为了简化Spring项目的搭建、开发、运行和部署的流程，更为了是帮助开发人员摆脱了原生Spring繁重的样板化配置 [19]。因此，近几年Spring Boot已经超越Spring 成为工业界最受欢迎的Java开发框架。本文的系统后端正是基于Spring Boot来开发完成的。

Spring Boot具备以下几个非常优秀的特性。首先，Spring Boot采用内置的Web容器Tomcat，相比于原生的Spring MVC我们部署时只需要项目编译打包生成的jar包，而无需配置Web容器的环境，这非常的方便于软件系统的迭代发布，大大减少部署的成本。其次，Spring Boot为Spring平台和很多第三方依赖库提供了开箱即用的设置和依赖支持 [20]，我们可以使用Spring Boot提供的约定优先配置，也可以通过自定义配置文件或者在启动命令中自己设置参数值来达到想要的配置效果，同时Spring Boot系列的各种starter启动器也简化了很多第三方依赖的maven配置，如Redis、RabbitMQ等，减轻了开发者配置第三方依赖的工作。然后，对于开发者来讲，Spring Boot提供了即时编译、热部署等工具包，更提供了安全监测、健康检查等内置的功能，使开发的效率得到提升。最后也是最关键的，近几年来微服务架构成为工业界非常火热的名词，越来越多的公司采用微服务架构来设计自己的项目，而Spring Boot支持Spring Cloud等微服务框架，其本身的特性也是针对于为微服务架构服务而设计的。

本文的所构建的系统为移动应用测试平台提供修复服务，需要使用轻量、对服务化架构支持度高的开发框架，Spring Boot的特性非常适合本系统，因此本文选择Spring Boot来构建后端程序。

2.2.2　Mybatis

Mybatis是Google开源的一款服务端数据持久层框架，能够支持定制化SQL与存储过程等功能。Mybatis通过XML配置文件和注解配置将Java中的POJO（Plain Ordinary Java Object）映射为数据库中的记录，从而消除了原生JDBC手工进行参数、代码的设置以及结果集的查询 [21]。本文系统的数据持久层正是采用Mybatis来进行开发完成的。

Mybatis具备以下几个优秀的特点。首先，Mybatis足够灵活，将SQL编写于XML配置文件中，易于管理，对程序代码和数据库设计没有侵入，没有影响。其次，提供DAO（Data Access Object）层，将数据访问和业务逻辑进行解

耦，使系统设计更清晰，提高了可维护性。Mybatis支持ORM（Object Relational Mapping）对象关系映射，为对象与数据库记录建立映射，便于查询结果集的获取以及数据的增删改。Mybatis提供的XML标签非常丰富，不仅有结果集等基础功能，更支持动态SQL如条件查询，支持数据分片、分库分表。Mybatis与Spring的集成非常方便，易于使用 [22]。

Mybatis从从功能上主要分为三个部分：API接口层、数据处理层、基础支持层。API接口层为开发人员提供了丰富的API，包括数据库操作、事务等，接口层接收到请求后会调用数据处理层来进行具体的数据处理。数据处理层则负责SQL的解析、执行和执行结果的处理，完成一次具体的数据库操作。基础支持层提供最基础的功能，包括数据库连接管理、事务管理、缓存管理和配置管理等组件，为数据处理层服务。

2.2.3 Dubbo

本文的所构建的系统为移动应用测试平台提供修复服务，采用面向服务架构（Service-Oriented Artitecture，SOA），需要轻量、集成效率高、可扩展性强的通信框架来实现服务调用。Dubbo是一个高性能的基于Java语言的开源的远程过程调用（Remote Procedure Call，RPC）框架，是阿里巴巴SOA服务化治理的核心框架，之后贡献给了Apache基金会成为顶级项目。而在工业界，Dubbo也是被广泛应用于分布式系统的开发，在各大互联网公司都受到高度欢迎。Dubbo致力于提供高性能、透明化的分布式远程调用解决方案。本文的系统是移动应用测试平台的子系统，通过RPC向移动应用测试平台提供服务，这里主要考虑到Dubbo传输效率高、可扩展性、无侵入式与Spring Boot集成高效等特点 [23]。

Dubbo的核心部分包括服务注册和发现、远程通讯、负载平衡、容错熔断等模块。如图 2.2所示，Dubbo的服务注册与发现是基于注册中心目录服务的自动注册与发现，服务消费者能够动态地查找服务提供者，服务地址透明化。对于开发者来，透明化的服务调用可以像调用本地方法一样调用远程方法，没有API的侵入。同时服务自动注册发现带来的好处是不再需要硬编码服务地址，添加和删除也更加平滑。Dubbo支持的注册中心有Zookeper、Redis、Multicast等，本文系统使用Zookeeper来作为注册中心，其本质是一个树状的文件存储系统 [24]，提供了对于数据节点的监听、更新、创建等API，对Dubbo服务注册和发现的支持非常好。Dubbo支持的通讯协议非常多，如RMI、HTTP、Hessian 等，默认使用单一长连接和NIO异步通讯，比较适合于数据量不大并发量高的传输服务，Dubbo对长连接和NIO进行了一层封装和抽象，提供了多种

线程模型、序列化方式和信息交换方式。Dubbo的集群容错提供了软负载均衡、地址路由、动态配置和熔断等机制，支持随机、轮询、一致性哈希、最少活跃等负载均衡算法 [25]。

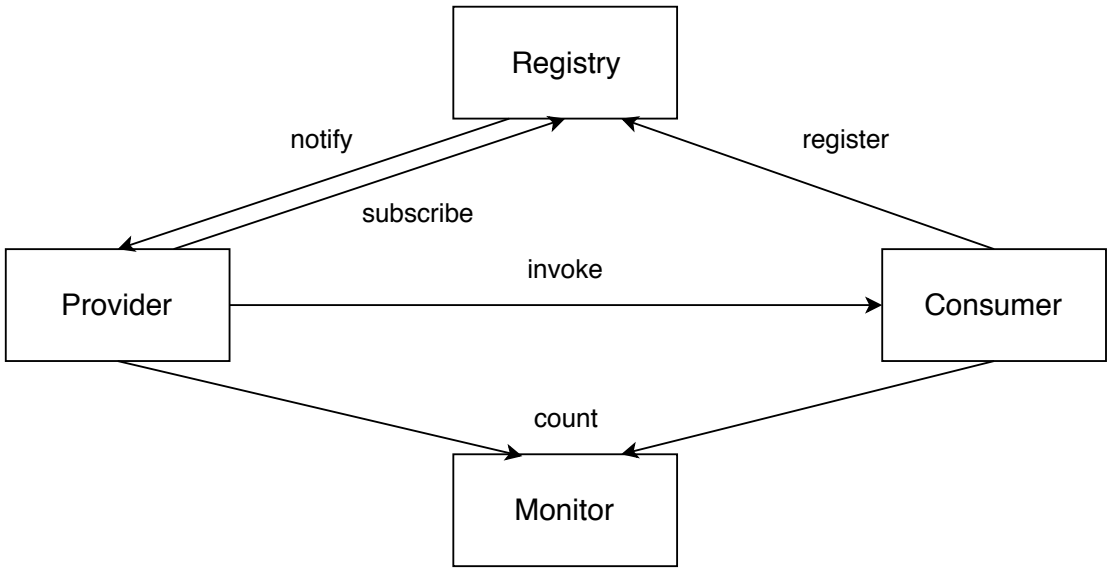

图 2.2: Dubbo服务注册和发现过程图

2.3　前端相关技术

2.3.1　Angular

Angular是Google团队推出的一款前端SPA（Simple Page Application，单页面应用）开发框架，体积小但功能强大，能够减轻前端开发的负担，提高Web开发的效率。Angular最初支持JavaScript，所以叫AngularJS，从第二个版本开始改名为Angular，也正式支持Typescript语言，支持移动应用的开发。对于后端开发者而言，Angular比业界其他几个流行的框架，如Vue、React等更友好，其代码风格和设计模式与后端开发语言和框架的风格有相当类似之处，尤其是引入了依赖注入和服务的概念，更易于后端开发者上手。本文系统采用基于TypeScript的Angular 4进行前端页面的开发，正式考虑其强大的功能和简便的开发方式 [26]。

Angular主要分为八个模块：模板、组件、模块、数据绑定、指令、元数据和依赖注入。模板是页面的具体内容，即HTML文件。组件是一块控制着视图的区域，可以当做HTML标签来使用。模块是一组模板和组件的组合，提供完

整的业务功能，往往也是业务划分的最小颗粒度。数据绑定是一种数据处理方式，定义了数据模型和视图之间的同步，而Angular提供了双向绑定。指令是包含指令元素的一个类，实现了向HTML中添加自定义的元素行为，动态地渲染页面。元数据则定义了如何处理一个类，如装饰器、模板和依赖提供者等。依赖注入是Angular为组件提供所需服务的方式。以上八个模块构成Angular全部的框架结构，也是完整渲染一个Web应用的生命构造 [27]。

Angular是符合MVVM（Model View View Model）设计模式的框架，比如其数据双向绑定的特性 [28]。如图 2.3所示，MVVM 模式引入了ViewModel来对视图和模型进行分离和解耦。主要原理是在ViewModel中构建一组状态数据，用来抽象视图的状态，然后通过数据的双向绑定来维持状态数据与视图的显示状态的一致，这样展示逻辑就可以通过状态数据来控制视图的状态。MVVM具备低耦合、可重用性、开发独立、可测试性等优势。而Angular的MVVM在标准模式上增加了Controller这个模块，主要作用是装饰、加工和处理ViewModel，包括ViewModel 的初始化、各种服务的组合、以及生命周期的控制。能够达到”分离关注点”的作用，使开发人员更关注于业务逻辑。

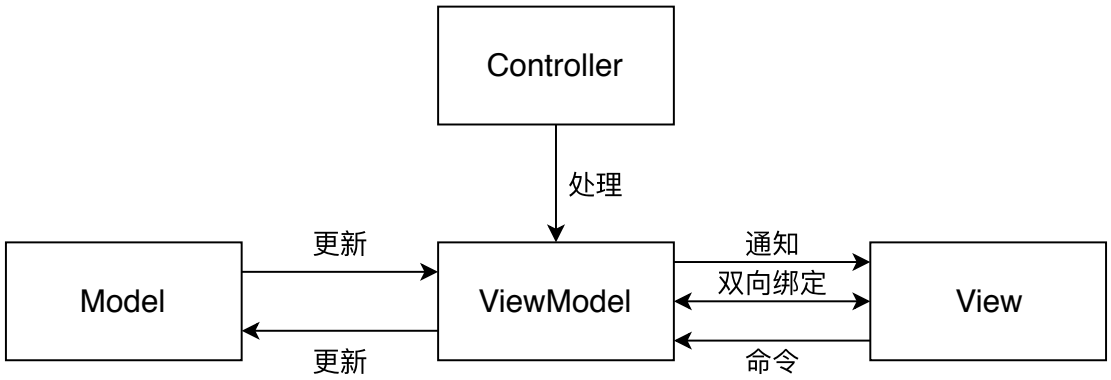

图 2.3: Angular的MVVM模式图

2.3.2　D3.js

本文的系统主要使用数据可视化技术来展示事件流图模型，将复杂的安卓GUI控件关系和事件逻辑形象地展示给测试人员，方便测试人员根据应用版本的变化来进行模型的编辑。D3.js是一个用于数据可视化的JavaScript函数库，全称为Data Driven Document。近年来，数据可视化变得非常流行，在很多报刊杂志、新闻媒体、门户网站等处得到广泛的运用。数据可视化可以把复杂而又庞大的数据变得易于理解，从而大大提高数据的可读性 [29]。

D3.js提供了各种各样简单易用又功能丰富的函数，大大降低了JavaScript直接操作数据的复杂度，减少数据可视化的工作量。从使用上来看，D3.js不需要安装，只需要引入一个文件便可以使用所有的功能，非常的方便高效。从功能上看，D3.js功能强大，我们使用D3.js能够轻易为图形添加可视化交互，定义图形的行为，其强大的可视化组件能够驱动DOM操作，而且支持使用HTML、CSS、SVG以及Canvas的形式展示数据 [30]。从运行上来讲，D3.js遵循现有的Web标准，无需借助任何其他框架就能独立运行在现代浏览器中，运行速度非常快，支持大数据集和动画。而没有框架限制就带来了另一个好处，使用者可以灵活运用自己的想法来表现数据。从数据源来讲，D3.js用于绑定DOM 的数据，可以是基本数据类型、对象数组、矩阵等 [31]。

本文主要使用了D3.js中的d3-force模块，此模块实现了用以模拟粒子物理运动的仿真模拟器 [32]，能够为一组节点创建仿真，并组合需要的力模型，并通过监听tick事件来不断更新图形。本文使用次模块来绘制具有编辑功能的有向图（Directed Graph Editor），即上文提到的事件流图模型。d3-force的运行过程如下，首先是处理图的节点，将节点数据导入d3中，每个节点按照一定的半径和旋转角度环绕，分别计算在x轴和y轴上的速度分量。然后为节点建立四叉树，四叉树的每个节点表示一个矩形区域，每个矩形区域分为四个部分，即该节点的四个子节点。按四叉树从上而下求所有节点的合坐标与合静电荷量，之后再进行电荷斥力的求解。然后是图的边的处理，先初始化连线，统计每个节点的度，求每一条边的起点度的占比，遍历所有连线，计算施加在连线两端节点的引力，最终推导出速度的变化，不断迭代计算，形成最终的布局。

2.4　其他技术

本文还使用了一些工具和算法，包括使用Fastjson来处理工具输出的JSON型数据，Fastjson是阿里巴巴推出的一款基于Java语言编写的数据解析序列化工具，它具备高性能、功能丰富、易于使用等特点，它为JSON数据操作提供了非常丰富的API，采用一种“假定有序快速匹配”的算法，把JSON解析的性能提升到了极致，因此在缓存序列化、Web交互、协议交互等有广泛的应用。

本文使用Druid来进行数据库连接池管理，Druid是阿里巴巴开源的一款数据库连接池实现，它即包含传统的C3P0、DBCP、PROXOOL等数据库连接池的优点，还增加了日志监控、SQL统计等功能，提高了数据库的实时监控和维护工作的效率。同时它与Spring Boot的集成也是非常简便易用。

本文在进行脚本修复算法的设计时借鉴了弗洛伊德最短路径算法，这是一种在具有权重的加权图中寻找最短路径的算法，它采用了动态规划的思想，通

过构建路径矩阵，从图的带权邻接矩阵开始递归进行N次更新，根据状态转移方程不断更新路径矩阵，最终得到所有两点间的最短路径距离，同时也可以引入后继节点矩阵来记录最短路径 [33]。

2.5　本章小结

本章节介绍项目中所使用到的一些技术、框架和工具。Appium是系统的底层核心，用于自动化遍历工具的实现和移动应用GUI测试脚本的运行，同时也是移动测试脚本的框架。Uiautomator是用于自动化遍历工具实时获取应用控件的数据。Spring Boot是用于系统后端的开发集成。Mybatis用于系统数据持久层的管理。Angular是用于系统前端的开发构建。D3.js是用于数据可视化具体是用于事件流图模型组件的构建开发。

第三章　安卓GUI测试脚本修复系统需求分析和设计

3.1　系统概述

在移动应用维护阶段，如果GUI发生更改，原有测试脚本将变得失效，因为GUI控件的相关属性和执行序列会发生变化，而原有的测试脚本编码的事件序列不再符合修改后的GUI结构。测试脚本的修复就是指将这些失效的测试脚本重新变得可用。目前解决这个问题只能依靠人工进行测试的维护来使这些不可用的测试重新变得可用。目前工业界大多数移动应用测试平台大多不具备测试脚本修复的相关功能，对于不同版本应用出现测试脚本失效的问题，大多数采用人工维护的方式，或者干脆雇佣众包人员重新编写，测试脚本的复用性非常的低 [34]。本系统的用户为企业的测试人员。本系统作为移动应用测试平台的子系统，为移动应用测试平台提供脚本修复的服务，从业务上进行补充，提高测试脚本的复用性，将人工直接审阅脚本变为基于事件流图模型的人机协同修复方式，大大减少了测试人员进行测试维护的成本。本系统从架构上提供可插拔式的独立服务，对移动应用众包测试平台无侵入 [4]。

从功能上来看，本系统分为应用管理模块、脚本管理模块、脚本运行模块、脚本修复模块。这四部分负责完成从应用的上传、脚本的上传、脚本的修复、脚本运行和运行结果收集等一系列业务流程。从修复流程上来看，本系统分为事件流模型生成模块、脚本映射模块、自动化修复模块和人机协同修复模块。这四部分构成了本系统的核心——安卓GUI测试脚本修复的全部过程。

从具体流程来看，测试人员上传应用和测试脚本到系统中，系统提供应用和脚本存储服务，并调用自动化遍历工具对应用进行扫描并解析出事件流图模型。用户也可以选择拷贝以前版本的测试脚本用来进行新版本应用的测试。用户可以在系统中查看指定版本应用的事件流图模型，同时用户可以在系统界面上对模型进行编辑修改，包括修改事件流图模型节点的相关信息，编辑模型节点之间的逻辑关系。之后用户选择重新运行修复后的测试脚本，系统根据修复后的事件流模型和模型脚本映射关系进行修复工作，重新生成测试脚本并实时调用手机服务运行修改后的脚本。测试脚本运行完毕后系统将运行结果即脚本成功率计算等数据后反馈给用户，用户根据运行结果不断重复上述步骤，不断地完善修复的效果。

3.2　需求分析

3.2.1　功能性需求

本系统的功能性需求包括上传应用、上传测试脚本、测试脚本拷贝、自动化修复、查看脚本运行结果、查看事件流图模型、人为干预修复等，详细描述如表 3.1所示，其中R4-R7是本系统的核心需求，优先级最高，也是构成修复系统的核心模块。

表 3.1: 功能性需求列表

需求编号	需求名称	需求描述	优先级
R1	上传应用	测试人员可以上传应用到系统中	低
R2	上传测试脚本	测试人员可以上传测试脚本到系统中	低
R3	测试脚本拷贝	测试人员可以拷贝旧版本的测试脚本用于新版本的应用的测试	低
R4	自动化修复	测试人员拷贝了老版本的测试脚本后，系统进行初步的修复	高
R5	查看脚本运行结果	测试人员可以查看当前版本测试脚本的运行结果	高
R6	查看事件流图模型	测试人员可以查看当前版本应用对应的事件流图模型	高
R7	人为干预修复	测试人员可以编辑事件流图，人为干预修复过程，来进行人工修复	高

3.2.2　非功能性需求

如表 3.2所示，为本系统的非功能性需求，主要包含可用性、可靠性、可扩展性、易用性和可维护性五个部分。

表 3.2: 非功能性需求列表

需求名称	需求描述
可用性	系统一个月内不能崩溃超过3次，崩溃后恢复时间不能超过60s
可靠性	系统执行的后台任务不会随着异常出现或实例崩溃而丢失，在实例恢复后应该在60s内恢复任务的执行
可扩展性	在出现新业务模块增加时或现有业务功能扩展务，无需修改现有的业务模块，能够在一周能实现新业务模块或新功能的开发
易用性	系统提供整洁友好、解释性高的图形用户界面，用户能够很快从导航中找到自己需要的功能，并能够根据系统的操作指引完成基本的业务流程
可维护性	系统实例可以随时上线或被销毁，被销毁的后台任务能够被上线的实例继续执行，保证系统实例的无状态性

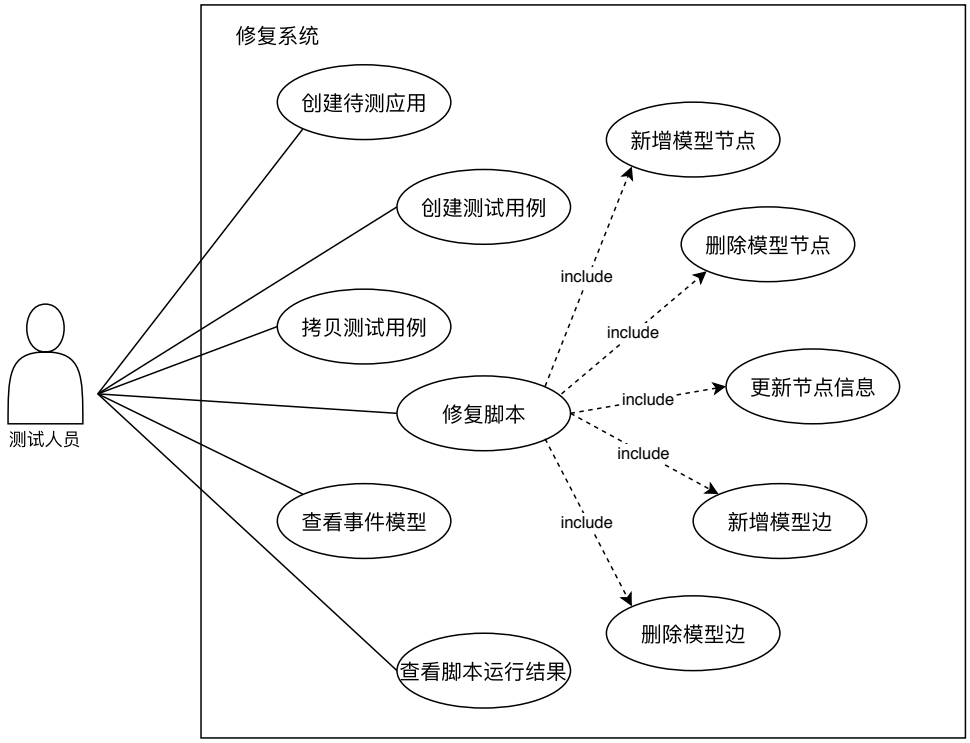

图 3.1: 系统用例图

3.2.3　用例描述

如图 3.1所示，是安卓GUI脚本修复系统的用例图，描述了测试人员在本系统中的用例。测试人员上传待测的应用和编写好的测试脚本到系统中，进行创建待测应用和测试用例。测试人员可以拷贝之前版本的测试用例用于新版本的应用。测试人员上传新版本的应用到系统中，之后测试人员就可以查看该版本和上一版本的事件流图模型。测试人员可以编辑该应用的事件流图模型并修改模型中事件结点对应的脚本代码。测试人员可以查看测试脚本运行结果和出错信息。表 3.3为对应的系统用例表，每个用例附有编号和名称，表中第三列为每个用例对应的功能性需求编号。

表 3.3: 系统用例表

用例编号	用例名称	对应需求编号
C1	创建待测应用	R1
C2	创建测试用例	R2
C3	拷贝测试用例	R3
C4	查看事件模型	R4
C5	查看运行结果	R5
C6	人工修复	R6、R7

以下为重要的系统用例的详细描述。

用例C1:创建待测应用

表 3.4: 创建待测应用用例描述

描述项	说明
用例名称	创建待测应用
参与者	测试用户
用例描述	测试用户在系统中创建待测应用
优先级	低
前置条件	登录成功并具备创建应用的权限
交互流程	1.测试用户进入创建应用界面，看到应用列表和版本列表，点击上传新版本应用按钮 2.选择待上传的应用，开始上传 3.系统显示上传成功后，填写应用相关信息 4.提交待测应用
后置条件	系统提示应用创建成功，用户在应用版本列表中可以看到新上传的应用并可以点击下载

用例C1描述的是测试用户在系统中创建待测应用的过程。系统对上传的待测应用文件提供存储和下载功能，用户可以下载自己上传的应用，系统允许同一应用存在多个版本。如果重复上传相同版本，系统会提示上传失败。

用例C2:创建测试用例

表 3.5: 创建测试用例用例描述

描述项	说明
用例名称	创建测试用例
参与者	测试用户
用例描述	测试用户在系统中创建测试用例并上传测试脚本
优先级	低
前置条件	登录成功并具备创建测试用例的权限限
交互流程	1.测试用户进入创建测试用例界面，看到测试脚本列表 2.填写用例的基本信息，如标题、测试步骤、前置后置条件 3.选择待上传的测试脚本，开始上传 4.系统显示上传成功后，提交测试用例
后置条件	系统提示测试脚本上传成功，测试用例创建成功，测试用例版本列表中可以看到新创建的测试用例并可以下载对应的测试脚本

用例C2描述的是测试用户创建测试用例的过程，这里的测试用例包括用例描述、测试步骤、和编写好的测试脚本，是本系统的核心业务实体。系统对上

传的测试脚本文件提供存储和下载功能，用户可以下载自己上传的测试脚本。系统将用户创建的测试用例显示在对应版本应用的测试用例列表中。

用例C3:拷贝测试用例

表 3.6: 拷贝测试用例用例描述

描述项	说明
用例名称	拷贝测试用例
参与者	测试用户
用例描述	测试用户拷贝以前版本的测试用例，用于新版本应用的测试
优先级	低
前置条件	登录成功并具备拷贝测试用例的权限，该应用存在多个版本，前版本应用存在测试用例
交互流程	1.测试用户进入拷贝测试用例界面，看到应用列表、版本列表和测试脚本列表，点击拷贝用例按钮 2.浏览版本列表，选择指定应用版本 3.浏该版本测试脚本列表，选择要拷贝的测试用例 4.点击确认按钮，提交拷贝
后置条件	系统提示测试用例拷贝成功，当前版本的测试用例列表可以看到拷贝的测试用例并下载该用例对应的测试脚本

用例C3描述的是测试用户拷贝测试用例的过程。对于旧版本的测试用例，系统提供测试用例拷贝功能，以将其复用于新版本应用上进行测试。系统将被拷贝的测试用例显示在新版本应用的测试用例列表中。

用例C4:查看事件模型

表 3.7: 查看事件模型用例描述

描述项	说明
用例名称	查看事件模型
参与者	测试用户
用例描述	测试用户查看指定版本应用的事件模型
优先级	高
前置条件	登陆成功并具备查看事件模型的权限，上传应用和对应版本的测试脚本或拷贝版本之前的测试脚本
交互流程	1.测试用户选择应用和版本 2.选择要查看的测试用例 3.进入测试脚本修复界面
后置条件	系统反馈该版本的应用所对应的事件流图模型

用例C4描述的是测试用户查看应用事件模型的过程。对于用户上传的待测应用，本系统提供为应用建立事件流模型和模型数据可视化的功能，用户在脚

本修复界面可以查看到该版本应用的事件流模型，包括节点信息即应用GUI事件相关数据。

用例C5:查看运行结果

表 3.8: 查看运行结果用例描述

描述项	说明
用例名称	查看运行结果
参与者	测试用户
用例描述	测试用户查看测试用例对应的测试脚本运行的结果
优先级	高
前置条件	登录成功并具备查看运行结果的权限，当前版本应用存在测试用例和与用例对应的测试脚本
交互流程	1.测试用户选择应用和版本 2.选择要查看的测试用例 3.进入测试脚本修复界面，点击运行结果按钮，进入脚本运行界面
后置条件	系统反馈该测试脚本的运行结果，包括成功率、出错日志信息等

用例C5描述的是测试用户查看测试脚本运行结果的过程。本系统提供真机运行测试脚本的功能，用户可以随时选择运行修复生成的测试脚本。本系统会对测试脚本的运行结果数据进行处理，并将日志输出、成功率等运行结果信息展示给用户。

用例C6:人工修复

表 3.9: 人工修复用例描述

描述项	说明
用例名称	人工修复
参与者	测试用户
用例描述	测试用户通过编辑事件流模型进行人工修复
优先级	高
前置条件	登录成功并具备查看事件模型和运行结果的权限，上传新版本的应用，并拷贝旧版本的测试用例
交互流程	1.测试用户选择应用和版本 2.进入脚本修复界面 3.查看事件流图模型 4.编辑事件流图模型及相关数据，包括图模型节点的创建、删除、修改，节点关系即图模型边的增加、删除等 5.选择运行修复后的测试脚本
后置条件	系统提示测试脚本修复运行中，修复工作与运行测试脚本完成后，在测试脚本运行结果界面能够看到运行结果

用例C6描述的是人工修复测试脚本的过程。系统不仅为指定版本的应用提供事件模型可视化功能，还通过可编辑图的方式为测试用户提供事件流图模型的编辑功能，用户可以人工进行模型的修复。

3.3　系统架构和模块设计

3.3.1　系统架构设计描述

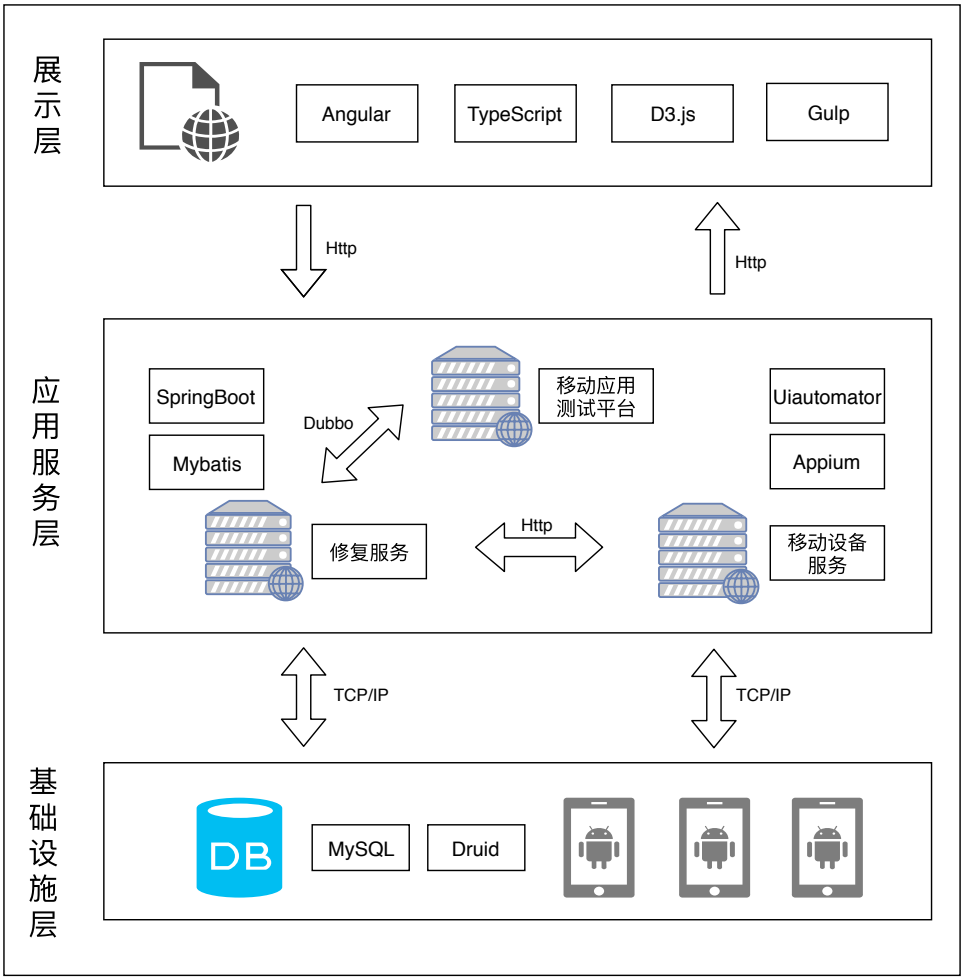

图 3.2: 系统架构图

如图 3.2所示，为安卓GUI测试脚本修复系统的总体架构设计图。从上到下分为三层，分别为展示层、应用服务层、基础设施层。本系统采用前后端分离的方式开发，并结合了SOA的软件架构模式。为了使前端统一风格，提高前端开发效率，本系统不具备前端，而是将前端集成在移动应用测试平台中。前端采用基于TypeScript语言的Angular框架进行开发，经过Gulp自动化构建工具打

包生成静态资源文件，使用D3.js来完成所有数据可视化的展示、人机协同修复的交互过程。修复服务后端采用基于Java语言的Spring Boot 框架进行开发，负责所有业务流程逻辑的实现、数据库检索和写入、应用和测试脚本文件的存储、修复算法的实现，并提供Restful的Http接口用来进行前后端交互。本系统为移动应用测试平台的子系统，为移动应用测试平台提供基于Dubbo框架的RPC服务，使用Zookeeper作为服务发现和注册中心，轻量、快捷而且可扩展性高。数据库采用MySQL 关系型数据库，数据库连接池采用Druid。自动化遍历工具和GUI测试脚本由移动应用设备服务提供，其运行基于Appium 和Uiautomator，分别负责安卓应用事件流图模型数据的生成和测试脚本的真机运行。

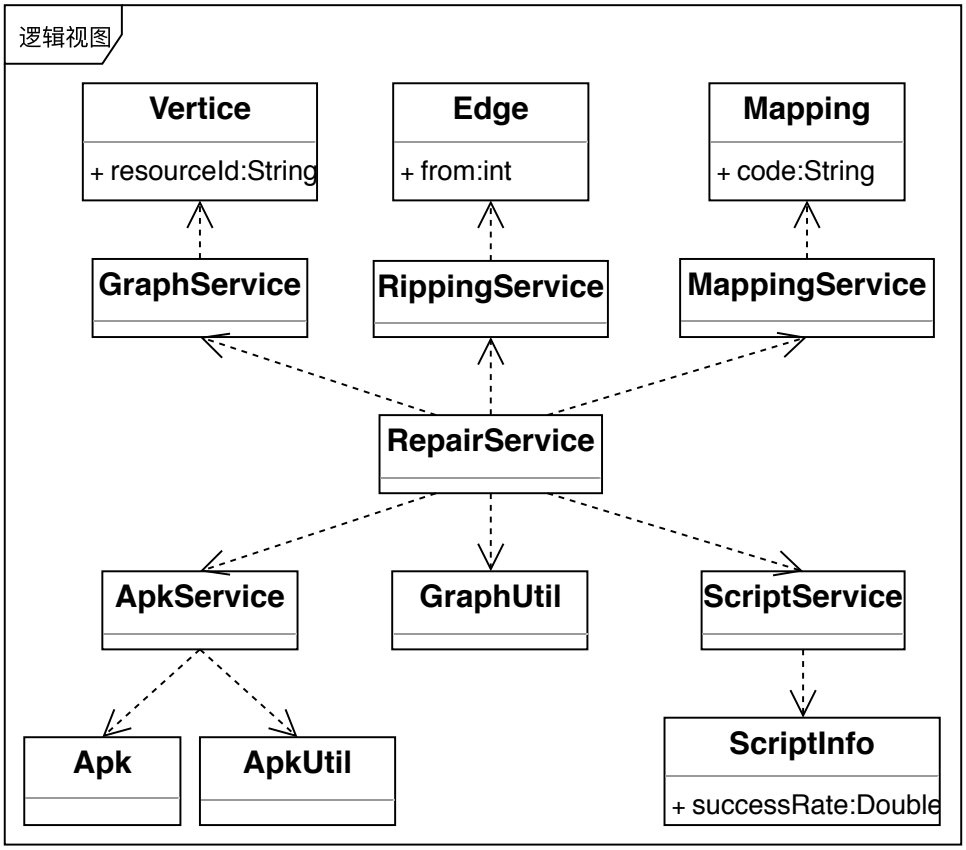

图 3.3: 系统逻辑视图

如图 3.3所示，为系统的逻辑视图，从最终用户视角来描述系统所提供的服务。整个系统由一系列抽象组成，GraphService主要提供应用的事件流图模型相关数据的查询和写入。RippingService主要提供为指定应用使用自动化遍历工具来建立事件流图模型的服务，包括自动化遍历工具的启动和对遍历结果输出的解析。MappingService主要提供为应用事件流图模型和测试脚本代码之间建立

映射关系的服务。ApkService主要提供应用上传和创建服务。ScriptService主要提供测试用例的创建、测试脚本拷贝和运行等服务。GraphUtil提供了图论最短路径算法服务。RepairService负责主要的测试脚本修复逻辑，并通过调用上述诸多服务模块来完成修复工作。

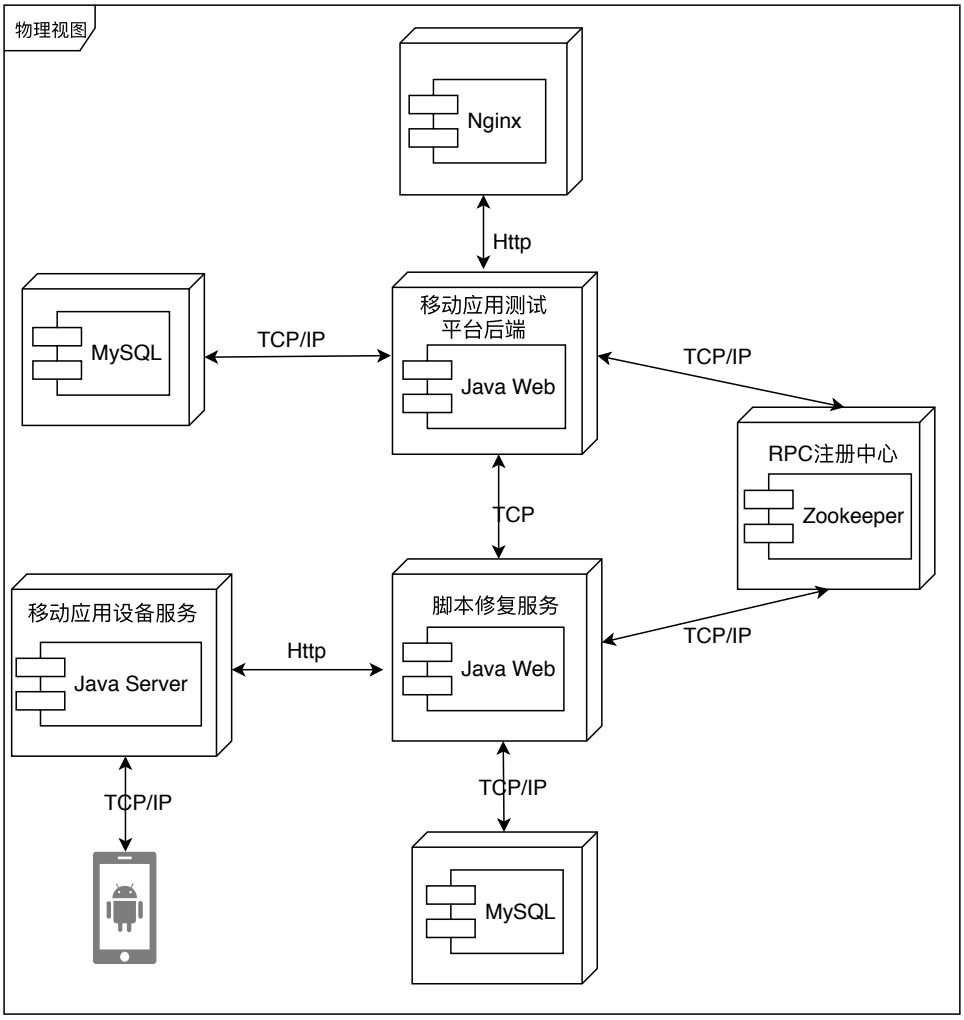

图 3.4: 系统物理视图

如图 3.4所示，为安卓GUI测试脚本修复系统的物理视图。本系统采用Nginx作为反向代理服务器，用于转发前端的Http请求，并且Nginx 还可以用作提供安卓应用和测试脚本下载的文件下载服务器，它能够将指定Url的请求转发到相应服务器的目录上。脚本修复服务的服务端是Java Web，部署在单独的云服务器上，通过Zookeeper来进行RPC的服务注册，提供相关的远程调用服务，而移动应用测试平台也可通过Zookeeper来进行服务发现。Zookeeper是阿里官方推荐的服务注册中心，且其功能和结构设计都非常有助于提高服务的可扩

展性、容错 [24]。脚本修复服务通过Http调用移动设备集群所提供的测试脚本运行服务。自动化遍历工具和运行脚本的移动应用测试服务也正是部署在移动应用设备集群上。移动应用测试服务直接连接移动设备，负责调用可用设备执行测试脚本，并返回测试脚本执行的执行结果和日志输出。

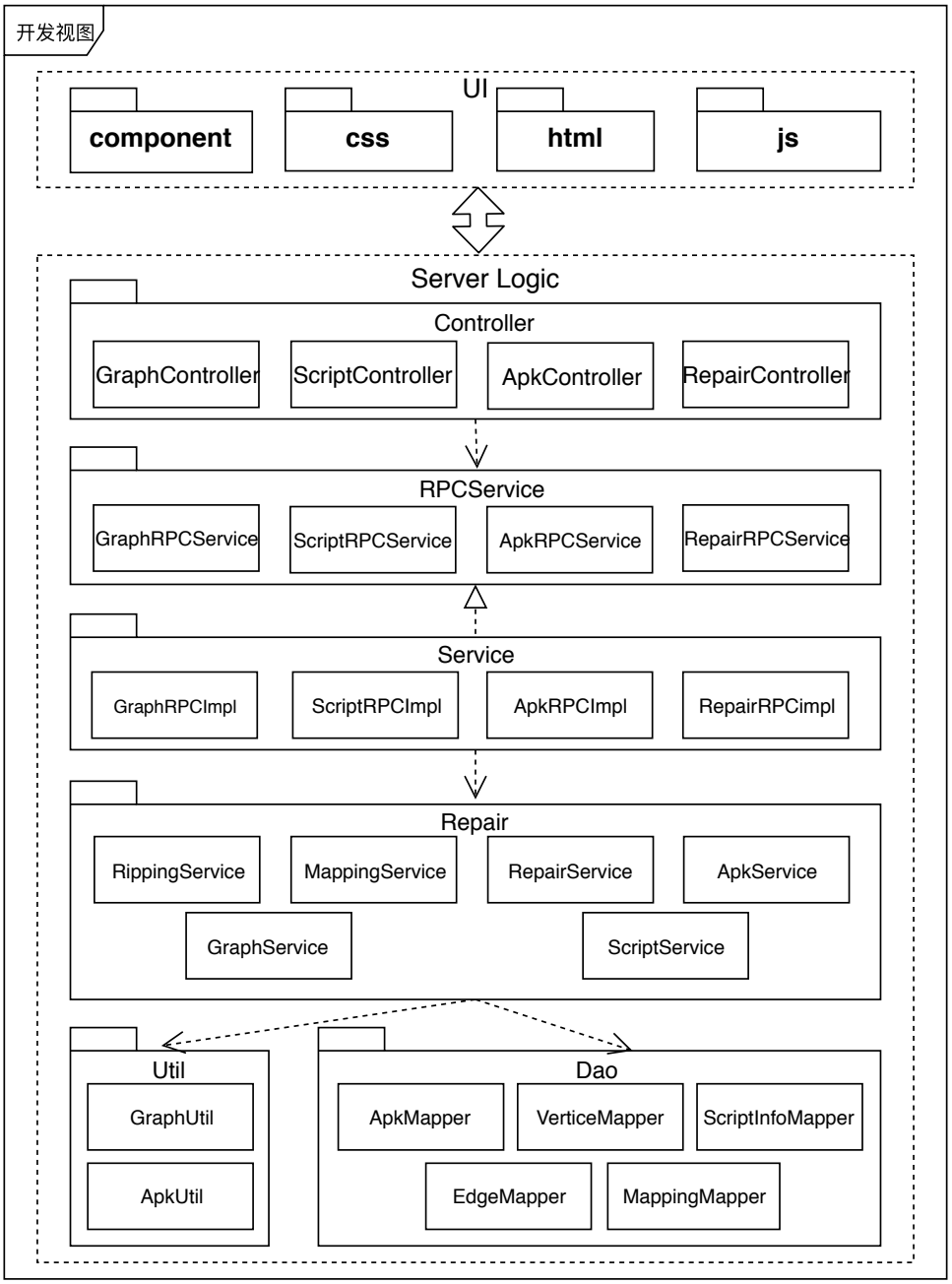

图 3.5: 系统开发视图

如图 3.5所示，为本系统的开发视图。本系统采用分层的体系结构进行开发。项目包结构分为两个部分一个是前端UI层，另一个是服务端逻辑层。UI层由四个部分组成，分别为component负责所有Angular组件的实现、css负责所有组件样式的实现、html负责所有组件页面的实现、js 负责所有第三方依赖。ServerLogic为服务端逻辑层，共由四个部分组成。Controller为控制器层由GraphController等组成，负责接受前端的业务请求和远程过程调用。RPCService为远程调用公共依赖，由GraphRPCService等组成，负责RPC业务接口的声明。Service层为远程调用服务提供层，是RPCService层的实现。负责修复业务层调用。RippingService等组成的Repair业务层为测试脚本修复工作的业务逻辑层，负责调用Util工具层和Dao数据访问层，实现具体的修复工作细节。

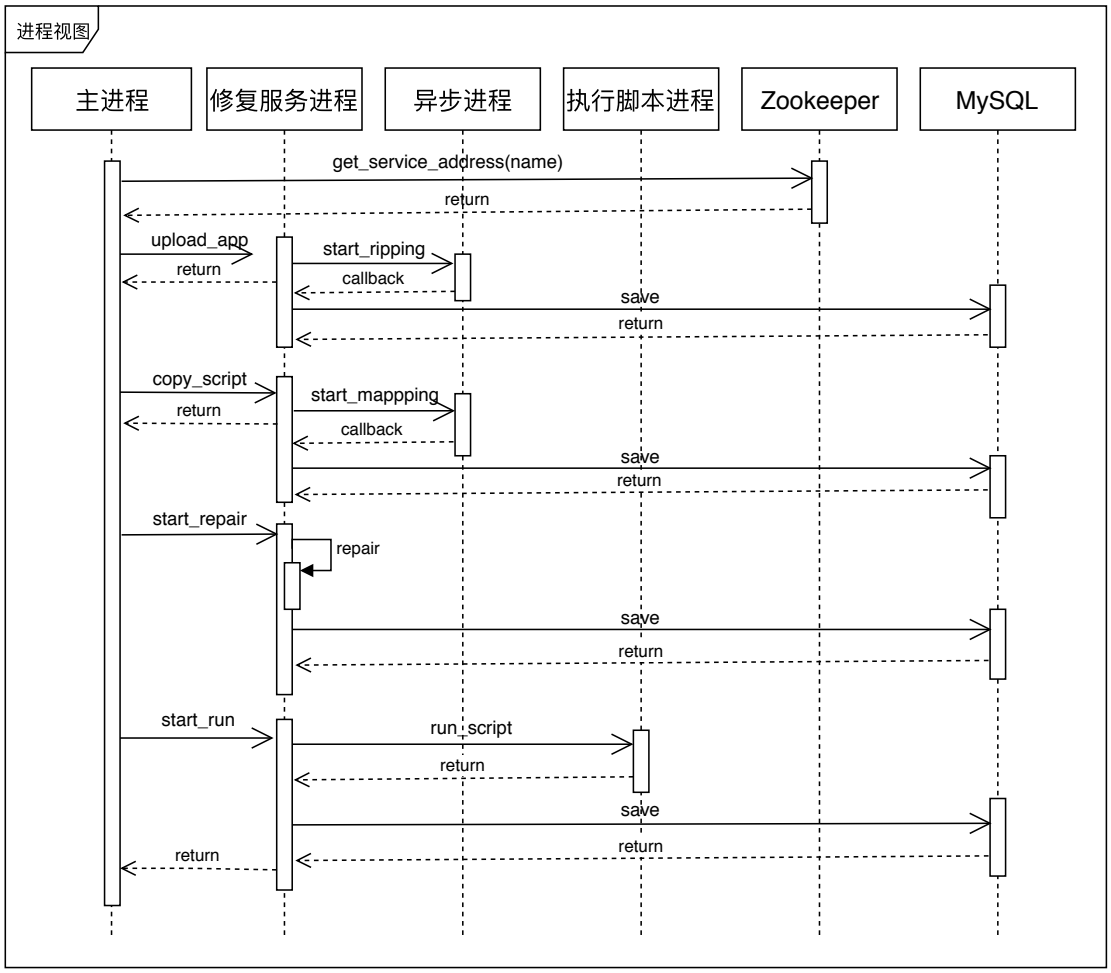

图 3.6: 系统进程视图

如图 3.6所示，为本系统的进程视图，从系统的集成视角描述系统相关进程、线程之间的通信关系。当系统需要访问修复服务时，移动应用测试平台主进程会向服务注册中心Zookeeper进程发送请求获取修复服务的IP地址等信息，根据返回结果进行远程过程调用。系统主进程同步调用修复服务进行待测应用的创建，修复进程接受请求后通过异步进程来启动自动化遍历工具开始Ripping工作，修复进程接受到异步进程的callback 后将事件模型数据落地到数据库中。系统主进程调用修复服务进行测试用例的拷贝，修复进程接受请求后通过异步进程来进行Mapping工作，接受到异步进程的callback后将映射数据落地到数据库中。系统主进程调用修复服务进行人工修复工作，修复进程在本地调用修复算法进行修复工作，并将结果落地到数据库中。系统主进程调用修复服务进行测试脚本运行，修复进程调用脚本执行服务进程运行测试脚本，收到结果后本地计算成功率并将数据落地到数据库中之后返回。

3.3.2　模块设计描述

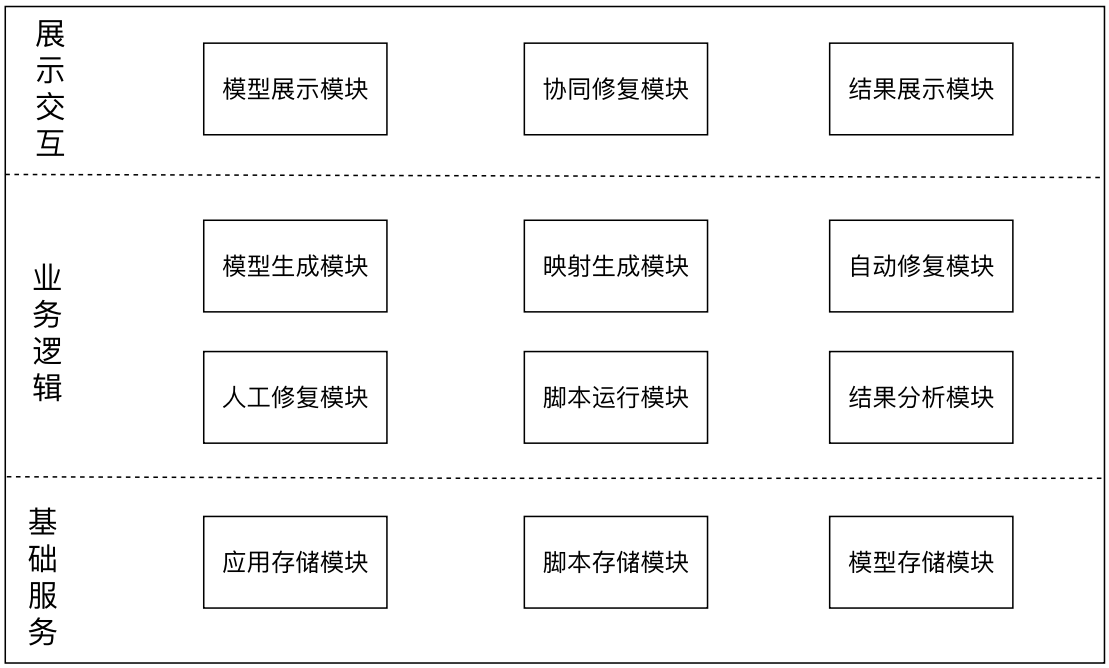

图 3.7: 系统模块设计图

如图 3.7所示，为安卓GUI测试脚本修复系统的模块设计图。分别由展示交互层、业务逻辑层和基础服务层组成。交互层主要由模型展示模块、协同修复模块、结果展示模块组成。业务逻辑层主要由模型生成模块、映射生成模块、

自动修复模块、人工修复模块、脚本运行模块和结果分析模块组成。基础服务层主要由应用存储模块、脚本存储模块、模型存储模块组成。

交互层主要负责本系统的数据展示和前端交互，包括安卓应用的事件流图模型展示，脚本运行结果的展示和人机协同修复的交互过程。在数据可视化领域，图结构数据的展示是非常常见的需求，本系统需要一个能够清晰直观地展现安卓应用GUI结构的图模型，图的节点为安卓GUI事件，也对应了一个安卓GUI控件，而图边则表示事件之间的逻辑关系，比如A->B，那么B节点对应的事件则发生在A节点之后。脚本运行结果的展示主要是当前修复生成的脚本运行的成功率，而这个成功率由测试代码中成功运行的行数除以总测试代码行数所计算得到，其次运行结果的展示还包括当前脚本运行的输出数据，包括运行日志和报错信息。

基础服务层主要提供底层的存储服务，主要包括数据库的读写和事务管理、应用和测试脚本的文件存储。包括应用的存储、版本管理、相关数据存储，脚本的存储、版本管理、脚本映射数据存储，模型数据的存储、版本管理。

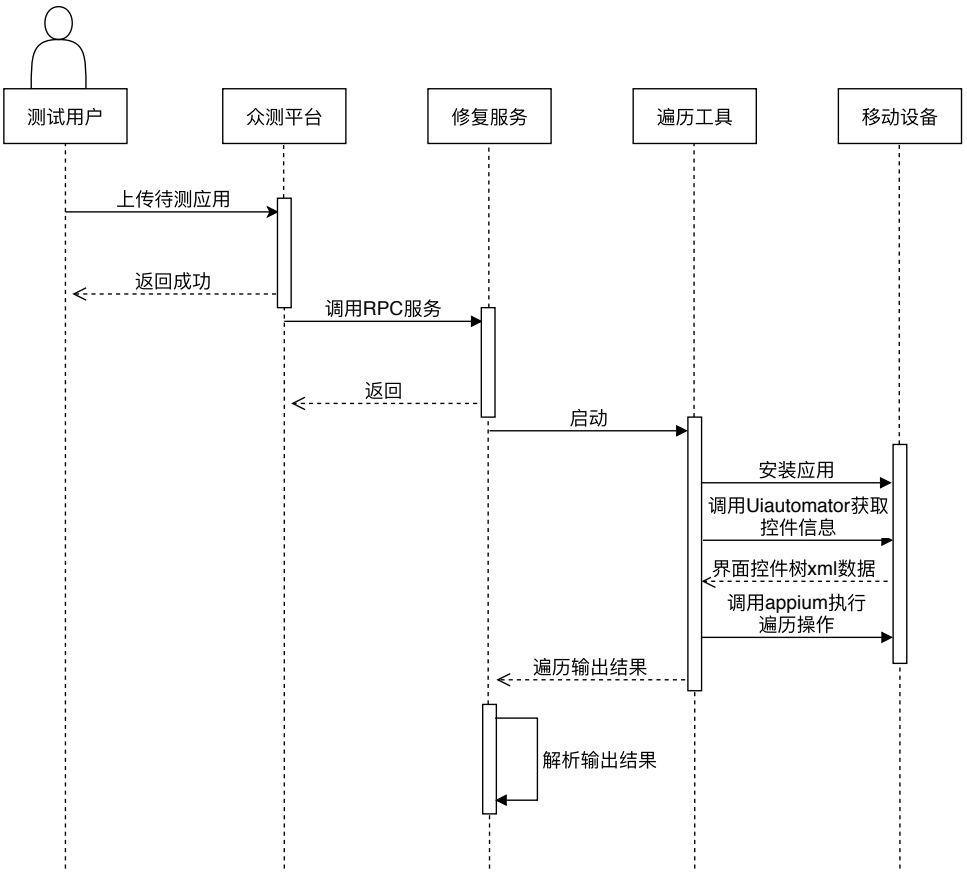

图 3.8: 事件模型生成模块时序图

模型生成模块：事件流图模型是整个系统的核心，也是测试脚本修复工作的基石。本系统采用自研的安卓自动化遍历工具对安卓应用进行扫描，该工具基于Appium和Uiautomator开发而成，使用深度优先策略遍历应用，该工具以JSON格式输出遍历过程中经过的每一个安卓GUI控件的相关数据，包括控件对应的各种属性，如：唯一标识、内容、控件类型、所属activity等 [35]，本模块再对工具的输出数据进行解析，提取其中的GUI控件数据和事件流关系，然后生成事件流模型的每一个节点和每一条边。如图 3.8所示，为事件流图模型生成的时序图。

具体流程如下：测试用户上传应用到移动应用测试平台系统中。移动应用测试平台调用RPC服务开始扫描应用。修复服务下载应用到本地并调用自动化遍历工具进行。工具调用安卓原生的adb服务安装应用到手机，调用Uiautomator获取应用界面控件树的xml数据，根据控件数据调用Appium执行对应的遍历操作。并将遍历过程中控件和遍历路径的数据输出到文件中。修复服务读取自动化遍历工具的输出文件，解析输出结果数据生成事件流图模型。

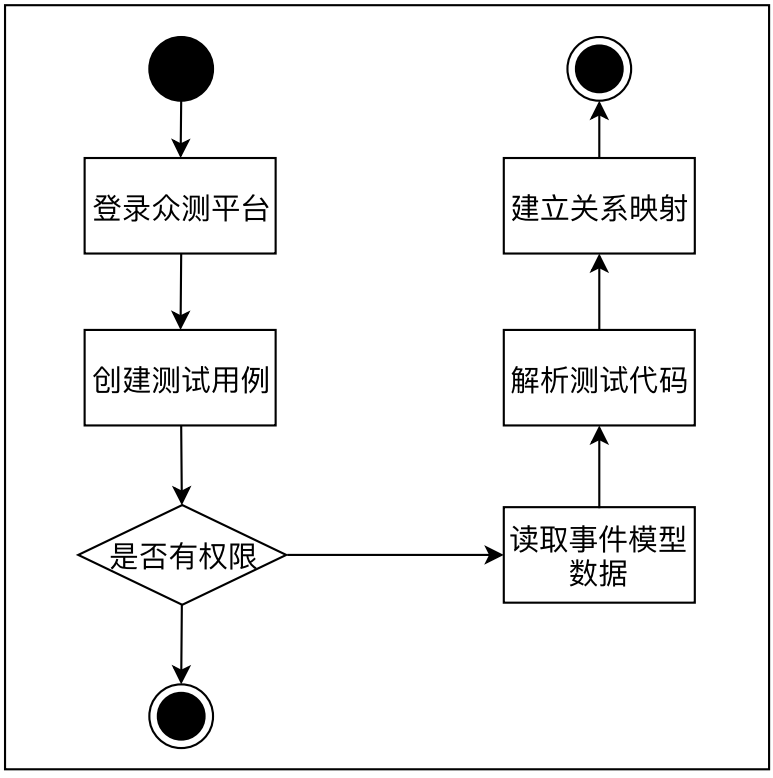

图 3.9: 映射生成模块流程图

映射生成模块：有了事件流图模型，我们就有测试脚本修复工作的基石，还需要将模型与测试脚本建立联系，进而通过模型来修复测试脚本。系统扫描用户创建测试用例时上传的测试脚本或拷贝测试用例的测试脚本，根据Appium框架的特点解析其中的测试代码，提取其中的用于定位元素的参数，再与事件流图模型建立映射，为后续的修复工作做准备。如图 3.9所示，为映射生成模块的流程图。

自动修复模块：自动修复模块是本系统的第二个核心，包含修复算法的实现、对于事件流图模型和模型脚本映射数据的处理过程以及新测试脚本的生成，本模块的执行流程如图 3.10所示。在接受到前端修复测试脚本的请求后，服务端先从数据库中读取事件流图模型和脚本模型映射数据，再调用自动修复算法进行修复工作，对所有建立映射关系的测试脚本代码进行检验，判断其对应的GUI事件节点执行和序列是否存在于当前版本的应用中，对那些失效的的节点映射或者执行序列进行对应的修复，最后生成新的测试脚本。具体的修复算法实现在下一章会做具体的描述。

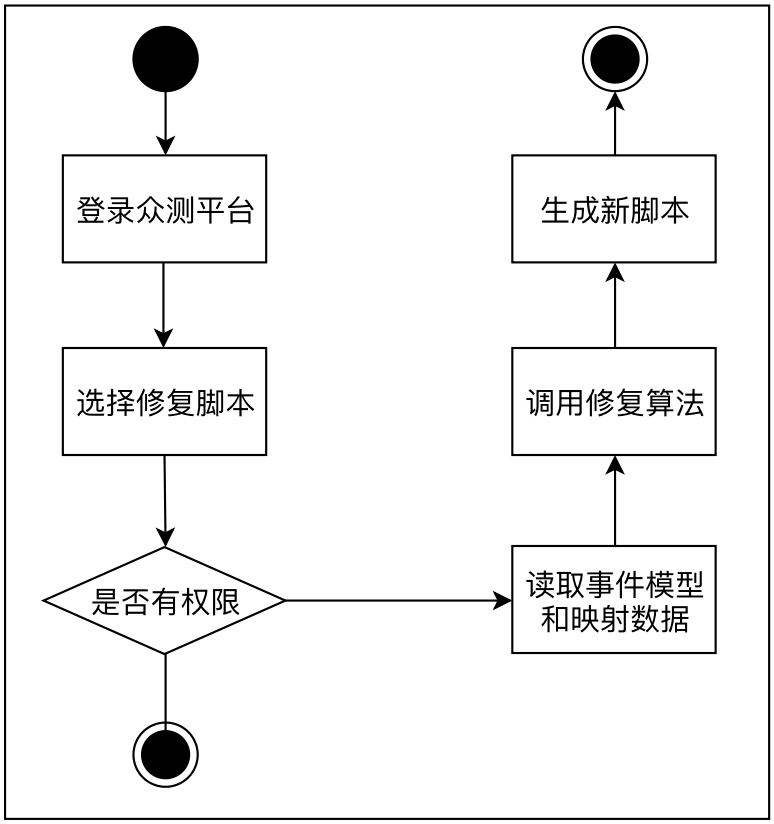

图 3.10: 自动修复模块流程图

人工修复模块：人工修复模块是本系统的第三个核心，主要实现的是人工干预修复，即测试用户对于事件流图模型的编辑行为。通过分析我们可以发现安卓应用版本的变化可分为以下几种情况，分别是新的GUI控件的增加、GUI控件相关属性的变动、原有GUI控件的删除和GUI控件之间的操作流的改变，以上几种GUI的变更情况可以分别对应到图模型中节点的创建和删除、节点数据的修改、节点关系即图模型边的增加和删除等。本模块主要负责接受前端关于上述GUI变更的数据，并根据对应的类型进行事件流图模型数据和模型脚本映射数据的更新，生成新的测试脚本。如图 3.11所示为人工修复模块的流程图。

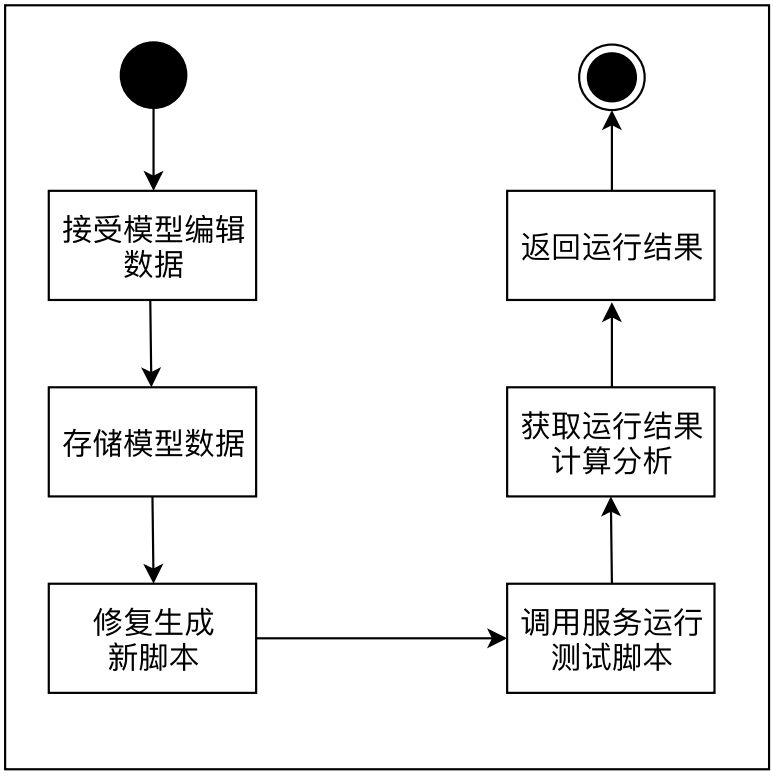

图 3.11: 人工修复模块流程图

脚本运行模块：本系统是采用人机协同的方式进行脚本修复，所以需要能够及时将修复生成的测试脚本在安卓设备上运行，获取运行的结果返回给前端，展示给测试用户。本模块负责调用移动设备服务运行指定的测试脚本并收集测试脚本的运行结果文件。如图 3.12 所示，为脚本运行模块的时序图。

主要流程如下:测试用户在移动应用测试平台系统中点击运行测试脚本按钮。移动应用测试平台调用RPC服务开始运行测试脚本。修复服务调用位于移动应用设备集群上的移动测试服务，传递测试脚本、待测apk和必要的数据。移动测试服务调用安卓原生的adb服务安装应用到手机，启动Appium 服务端，调

用Appium的服务执行测试脚本，并将运行日志数据返回给修复服务。修复服务解析测试脚本运行的日志输出，计算成功率等相关结果。

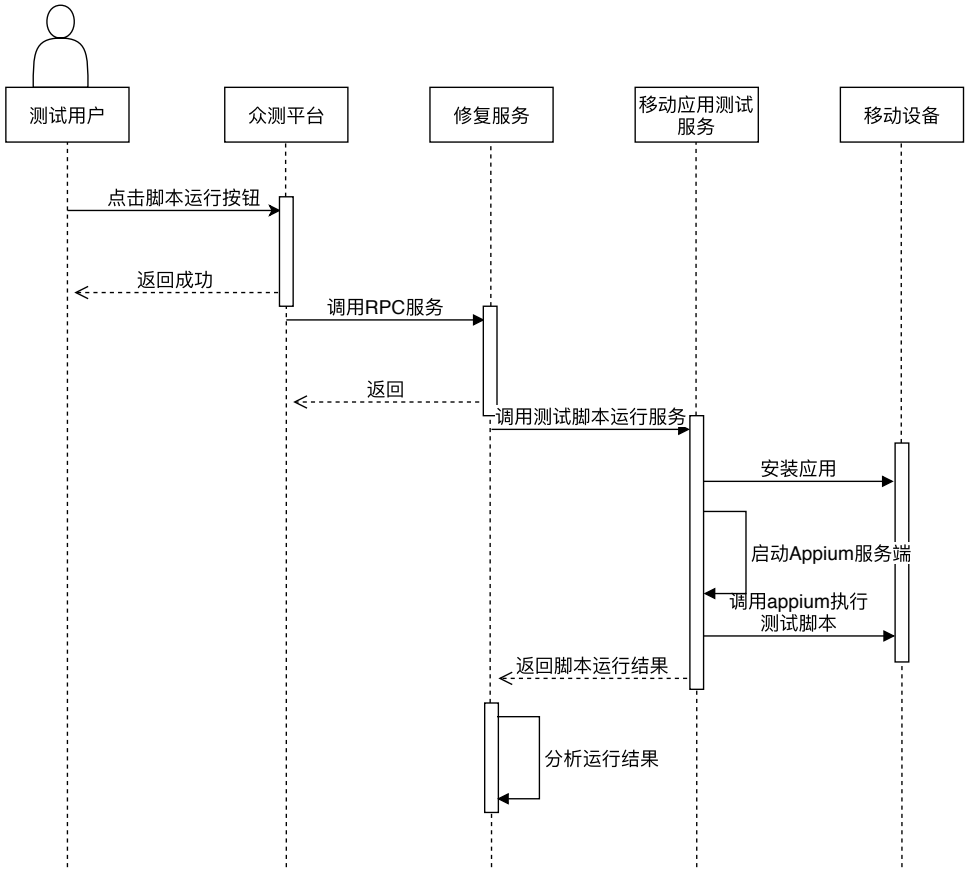

图 3.12: 测试脚本运行模块时序图

结果分析模块：本模块负责的是处理分析移动应用测试服务调用真机运行测试脚本之后返回的结果，主要是处理测试脚本运行输出的日志，包括出错信息等数据。本模块负责从日志中提取相关的数据来计算脚本运行的成功率以及各评估指数等。

3.4　数据库设计

如图 3.13所示，为安卓GUI测试脚本修复系统的数据库实体关系图的主要部分。包括五个实体，分别为应用本身、事件流模型节点、事件流模型边、运行的测试脚本、模型脚本映射。应用实体是整个系统的主要业务实体，它的唯一标识和版本号是其余业务实体的直接外键。测试脚本是测试人员在系统中为指定版本应用所创建的测试用例，所以待测应用与测试脚本是一对多的关系。

测试人员上传新版本应用后，系统调用自动化遍历工具对该应用进行分析，生成事件流模型节点、事件流模型边，因此应用与事件流模型节点和边均为一对多的关系。有了事件流图模型的数据，系统会扫描所有的测试脚本，解析其中的测试代码，分别为每一句测试代码建立与事件流图模型节点的映射关系。考虑到安卓GUI的特性，可能测试脚本的事件序列会重复经过同一个事件，所以事件流图模型节点与模型脚本映射之间为一对多的关系。而这些实体中比较核心的便是事件流图模型节点和模型脚本映射。

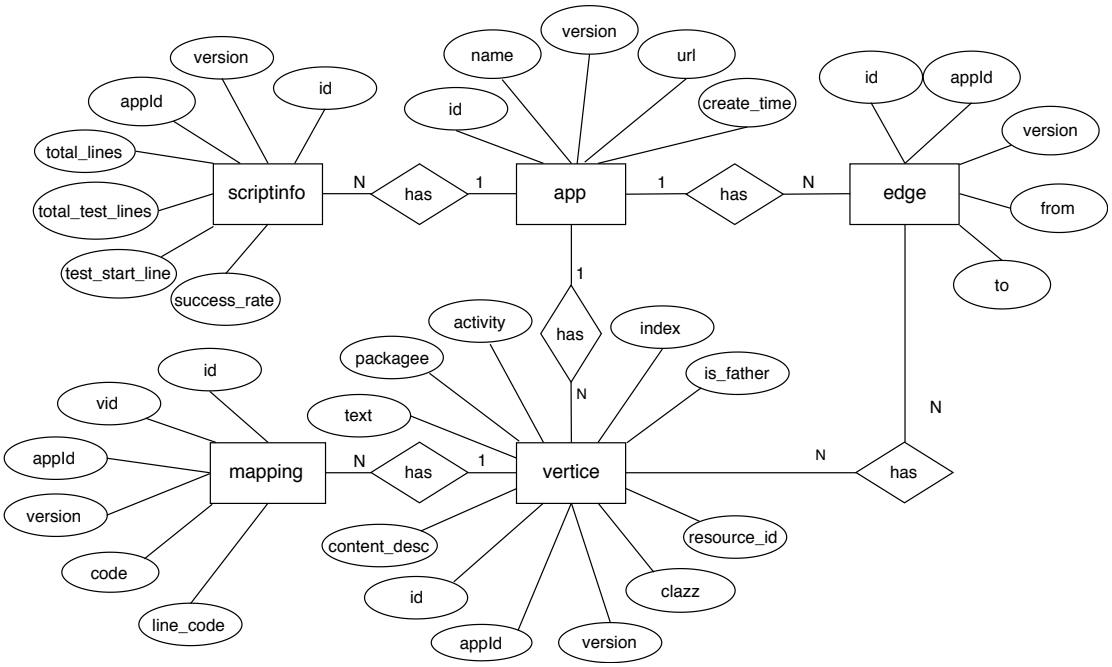

图 3.13: 实体关系图

模型节点数据结构如表 3.10所示，主要包含了模型中该节点所对应的事件涉及的安卓控件的数据信息，如控件类型、控件资源标识、安卓应用包名、控件标签、控件内容、控件所属activity、控件组序号等。考虑到移动应用测试框架Appium的代码特点，这些数据都是编写自动化测试脚本时用于捕捉和定位控件的，必须存入数据库中。

事件流模型边表存储的是事件流模型中事件之间的逻辑关系，即事件流图中的边数据，包括对应应用标识、对应应用版本、起始节点编号、指向节点编号等字段，与事件流节点表构建了完整的事件流图模型，为前端关于事件流图模型展示提供数据存储。

表 3.10: 事件流图模型节点表

字段	字段名称	数据类型	可否为空	描述
id	唯一标识	int	否	主键
appid	应用标识	int	否	外键，用于检索
version	版本号	varchar(32)	否	外键，用于检索
clazz	类名	varchar(255)	是	安卓控件的类名
resourceid	控件标识	varchar(255)	是	安卓控件的资源唯一标识
packagee	包名	varchar(255)	是	安卓应用的包名
text	控件标签	varchar(255)	是	安卓控件的标签数据
activity	控件所属页面	varchar(255)	是	安卓控件所在的页面，所属的（activity）
contentdesc	控件内容	varchar(255)	是	安卓控件的内容数据
index	控件序号	int	否	安卓控件在控件组中的序号，单个控件为0
isfather	是否为父组件	bit	否	安卓控件是否为父组件即是否包含子组件

模型脚本映射的数据结构如表 3.11所示，主要包括对应应用标识、对应应用版本、对应模型节点标识、测试代码、测试脚本行号。模型脚本的映射关系主要用于后续的脚本修复，测试代码和脚本行号用于测试脚本的生成，对应应用标识和对应应用版本用于系统对模型脚本映射数据的检索。

表 3.11: 模型脚本映射表

字段	字段名称	数据类型	可否为空	描述
id	唯一标识	int	否	主键
app_id	应用标识	int	否	外键，用于检索
version	版本号	varchar(32)	否	外键，用于检索
vid	节点编号	int	是	测试脚本语句对应的事件模型节点编号
code	代码语句	varchar(255)	否	映射对应的测试脚本语句代码
line_code	脚本行号	int	否	映射对应的测试脚本语句行号

表 3.12: 脚本信息结构表

字段	字段名称	数据类型	可否为空	描述
id	唯一标识	int	否	主键
appId	应用标识	int	否	外键，用于检索
version	版本号	varchar(32)	否	外键，用于检索
total_lines	代码总行数	int	否	测试脚本总代码行数
total_test_lines	测试代码总行数	int	否	测试脚本测试代码总行数
test_start_line	测试代码起始行号	int	否	测试脚本测试代码起始行号
success_rate	成功率	double	否	成功率

脚本信息表的结构如表 3.12所示,存储的是指定版本应用的测试脚本相关信息，包括对应应用标识、对应应用版本、测试代码行数、代码总行数、测试代码起始行号、运行成功率等字段，该表中的数据为系统经过计算所得，用于前端关于测试脚本运行结果的展示。

应用信息表存储的是各版本应用的相关信息数据，包括在主平台的唯一标识、版本号、存储位置、应用名称、创建事件等。

3.5　本章小结

本章描述了安卓GUI测试脚本修复系统的整体概述，主要分析了修复系统的功能性需求和系统用例设计，然后详细介绍了系统的总体架构设计思路、功能模块的划分思路、六个核心模块的具体功能，通过时序图、流程图等工程分析方法仔细阐述了每个模块的设计思路。最后介绍本系统的数据库结构和设计思路。本章重点阐述了脚本修复过程的设计方案，分析了脚本修复的具体流程。通过需求和系统流程的分析，确定总体的设计方案，为下一章的详细实现打下基础。

第四章　安卓GUI测试脚本修复系统详细设计和实现

4.1　事件模型生成模块

4.1.1　模型生成模块概述

安卓GUI测试脚本修复系统的核心是测试脚本的修复功能，但是一切修复工作都是建立在应用的事件流图模型上，所以为应用生成事件流图模型是我们研究工作的第一步，也是本系统业务流程的基石。事件流图模型由两部分组成，一部分是安卓应用GUI事件，在事件流图模型中体现为图模型的节点（vertice），它主要由GUI事件发生时所对应的GUI元素的数据组成 [36]，具体的数据内容如表 4.1所示，在这里我们根据Appium 定位元素的参数选择其中的部分信息存储到数据库中来构建节点。另一部分是安卓应用GUI事件的逻辑关系，在事件流图模型中体现为图模型的边。所以我们需要获取当前应用所有的GUI控件数据和事件执行序列即可成功构建出该应用的事件流图模型。

表 4.1: 安卓GUI元素属性表

属性	描述
resourceId	GUI元素对应的资源唯一标识
text	GUI元素的文本信息
class	GUI元素的类型信息，如：android.widget.TextView
package	该应用的包名
contentDesc	GUI元素的描述信息
index	GUI元素在本界面上同一层次组件中的序号
clickable	GUI元素是否可以被点击
checkable	GUI元素是否可以被check
enable	GUI元素功能是否可用
checked	GUI元素是否被check
longClickable	GUI元素是否可以被长按
focusable	GUI元素是否可以被屏幕聚焦
focused	GUI元素是否被屏幕聚焦
selected	GUI元素是否被选中
bounds	GUI元素所处区域的坐标

4.1.2　模型生成模块实现

本文借助于逆向工程的思想，使用自研的自动化遍历工具，对目标应用进行自动化遍历，工具的遍历过程如下：首先检查对应的安卓应用文件是否

存在，并通过安卓原生的adb命令检查目标设备是否处于在线状态，然后启动Appium服务端，调用adb命令向目标设备安装指定的安卓应用，之后进行应用界面的深度遍历。遍历前先进行安卓权限询问窗口的处理，这里是因为安卓原生的授权组件并不属于目标应用，之后调用Uiautomator提供的API获取到当前界面GUI控件布局的xml数据，解析这些xml数据获取其中可以点击的控件列表，然后遍历当前的控件列表，点击控件并检查界面hash值是否发生变化，如果变了说明此操作使得界面增加了GUI控件，重新获取xml数据更新控件列表，还需要检查界面activity是否发生变化，如果发生变化则说明发生了页面跳转，亦重新获取xml数据更新控件列表，并按照深度优先遍历的策略继续遍历 [37]。

模型生成模块负责调用自动化遍历工具获得结果文件后，通过对工具的遍历结果进行解析，提取其中能够用于构建事件流图模型的GUI控件数据和序列关系，根据提取结果来构建事件流图模型。

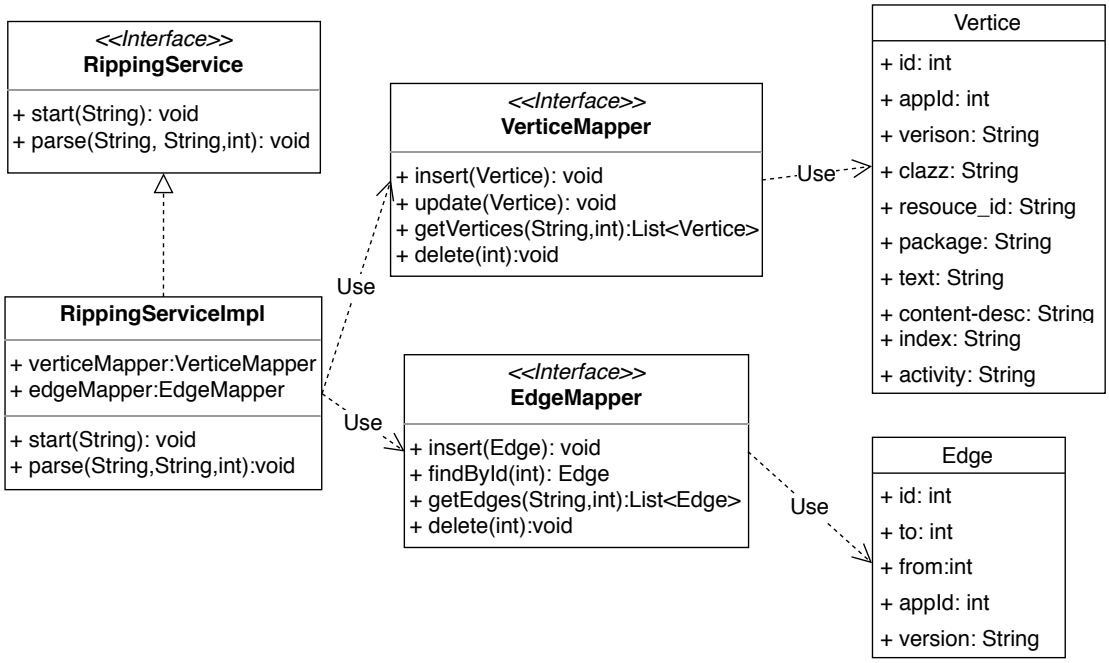

图 4.1: 模型生成模块的设计类图

模型生成模块的设计类图如图 4.1所示，Vertice和Edge封装用于构建事件流图模型的事件节点和边的数据结构，VerticeMapper和EdgeMapper提供了数据访问层（DAO）的各种服务，负责数据库的读写，RippingService封装了生成事件流图模型的各业务步骤，包括启动自动化遍历工具、解析工具的输出文件、建立事件流图模型等。

自动化遍历工具的启动由RippingService.java提供，这里首先判断该应用文件是否存在，不存在则抛出无法找到文件的异常，然后通过Java语言原生的java.lang.Runtime运行启动自动化遍历工具的shell命令，并获取执行线程的输出，记录到相应的日志文件中，由于这一步执行的等待时间可能比较长，本文采用异步线程的方式来执行此步骤。具体的代码如图 4.2所示。

```
public void start(String apkName, String version, String appId) throws Exception{
        File apk = new File(apkDir + apkName);
        // …检验文件是否存在
        DateFormat format = new SimpleDateFormat("yymmdd");
        String taskId = format.format(new Date());
        cmd = String.format(cmd, apkName, taskId, deviceId);
        try {
            Process process = Runtime.getRuntime().exec(cmd, null, new File(dir));
            StringBuffer resStr = new StringBuffer();
            InputStream stream = process.getInputStream();
            Reader reader = new InputStreamReader(stream);
            BufferedReader bReader = new BufferedReader(reader);
            // … 根据输出结果判断工具是否运行成功
            parse(apkName, version, appId);
            // … 输出日志文件
            reader.close();
            bReader.close();
            process.getOutputStream().close();
        } catch (IOException e) {
            // … 处理异常
        }
}
```

图 4.2: 启动自动化遍历工具代码

系统还需要对工具的输出数据进行解析，提取其中的事件流，包括安卓GUI控件的数据和事件序列，从而构建事件流图模型。这部分业务也由RippingService.java提供，自动化遍历工具的输出文件内容为JSON格式数据，这里采用阿里巴巴开源的Fastjson序列化工具，读取每一个事件的输出数据，再将其转化为图结构，提取GUI控件属性数据构成事件流图模型的节点，并根据输出序列和父子组件关系构成事件流图模型的边，具体的代码如图 4.3所示。

4.1.3　事件模型准确度评估

本文对于模型准确度的评估通过实验人工审核统计，通过AndroidSDK内置工具Uiautomatorviewer可以获取应用每个界面的xml布局信息，从xml布局

```
public void parse(String apkName, String version, Long appId){
        DateFormat format = new SimpleDateFormat("yyyyMMdd");
        String actionFile = deviceId + "_" + format.format(new Date());
        File file = new File(dir + "/TestAction/" + actionFile + "_action.txt");
        List<JSONEvent> events = new ArrayList<>();
        try{
            FileInputStream inputStream = new FileInputStream(file);
            BufferedReader reader =
                    new BufferedReader(new InputStreamReader(inputStream));
            for (String res; (res = reader.readLine()) != null;) {
                JSONEvent event = JSON.parseObject(res, JSONEvent.class);
                if (event.getComponent() != null) {
                    events.add(event);
                }
            }
        }catch (FileNotFoundException e){
            // … 处理异常
        } catch (IOException e){
            // … 处理异常
        } catch (JSONException e){
            // … 处理异常
        }
        // 将事件数据转化为图结构
        convertToGraph(events, version, appId);
    }
```

图 4.3: 解析工具输出数据代码

信息中我们可以获取到当前界面上所有的GUI元素，包括可以产生点击事件的GUI控件。通过统计这些数据，我们可以得到一张有人工绘制而成的事件流图模型，再与模型生成模块得到图模型对比，便可得出图模型的准确度。具体的实验过程和结果数据将在下一章展示。

4.2　脚本映射模块

4.2.1　脚本映射模块概述

事件模型生成模块为目标应用建立了事件流图模型，系统还需要建立起事件流图模型和测试脚本的映射关系，后续的脚本修复工作需要基于二者的映射关系来进行。映射关系由两部分组成，一部分是测试脚本本身的信息，包括定位控件和执行事件的代码、行号等数据，用于后续生成修复的测试脚本。另一部分则是测试代码对应的模型节点编号，用于后续的修复工作。本模块通过对Appium测试框架代码风格的总结，使用正则匹配的方式进行测试代码的分

析，并通过提取测试代码中定位GUI控件的参数来检索与之对应的事件流图模型节点，从而完成建立映射关系的业务流程。

4.2.2　脚本映射模块实现

本模块通过匹配测试脚本文件中的测试代码和数据库存储的事件流图模型数据来完成脚本模型的映射。如 4.4所示，为脚本映射模块的设计类图。Mapping和ScriptInfo类封装了模型映射关系和测试脚本相关信息的数据。MappingMapper和ScriptInfoMapper提供了数据访问层（DAO）的各种服务，负责数据库的读写，MappingppingService封装了建立事件模型脚本映射关系的业务步骤，包括测试脚本代码的解析和分类、不同定位方式测试代码参数的提取和事件流图模型节点的检索等，VerticeMapper、Vertice等类在上一章已经描述过，这里就不再赘述了。

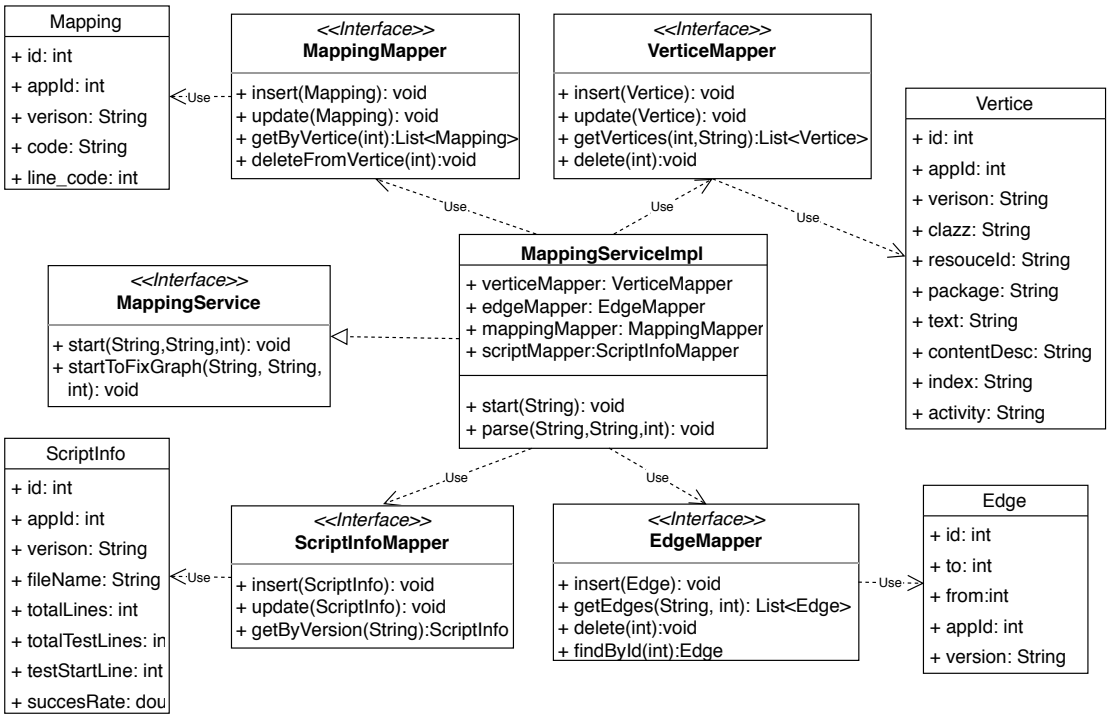

图 4.4: 脚本映射模块设计类图

如表 4.2所示，为Appium测试框架支持的定位安卓界面GUI控件的方式，其中比较常用的控件定位方式是使用Xpath、Id、Name、Class来进行定位等。经过调查平台现有的用户所编写的测试脚本，这其中使用Xpath来定位控件的方式最为普遍，而且其支持参数也更多，Xpath 定位方式支持使用绝对位置、text、

conten-desc等多种参数来进行GUI控件的定位。本模块基于Appium测试框架的定位方式来提取测试脚本中的参数，从而进行映射生成工作。

表 4.2: Appium定位控件方式

定位策略	解释	对应测试代码
Xpath	使用xpath绝对位置进行定位	findElementByXpath
Id	使用控件的resouceId来进行定位	findElementById
Name	使用控件的text属性来进行定位	findElementByName
Class	使用控件的class进行定位	findElementsByClassName
Accessibility id	使用控件的contentDesc进行定位	findElementByAccessibilityID
Android uiautomator	使用uiautomator相关语法进行定位	findElementByAndroidUIAutomator

参考表 4.2，本文根据Appium的测试代码风格归纳总结出了八个正则表达式，如图 4.5所示，用于匹配移动应用测试脚本中的测试语句。本模块根据正则表达式判断测试代码所使用的控件定位方，再根据不同的定位方式提取出其中用于定位GUI控件的参数，最后从该应用的事件流图模型数据中检索出与之对应的图模型节点，从而建立事件流图模型与测试脚本的映射。

```
public class CodePattern {
    public static final String POSTION_BY_ID =
            "^\\s*driver.findElementById\\(\".*:id/.*\"\\).click\\(\\);$";
    public static final String POSTION_BY_INDEX =
            "^\\s*driver.findElementByXPath\\(\"//.*" +
                    "\\[@index='[0-9]']\"\\).click\\(\\);$";
    public static final String POSTION_BY_TEXT =
            "^\\s*driver.findElementByXPath\\(\"//.*" +
                    "\\[@text='.*']\"\\).click\\(\\);$";
    public static final String POSTION_BY_CONTENTDESC =
            "^\\s*driver.findElementByXPath" +
                    "\\(\"//.*\\[@content-desc='.*']\"\\).click\\(\\);$";
    public static final String WAIT_FOR_ACTIVITY =
            "^\\s*waitForActivity\\(driver,\\s*\".*\"\\);$";
    public static final String SWIPE =
            "^\\s*driver.swipe\\(.*,.*,.*,.*,.*\\);$";
    public static final String SENDKEYS_BY_ID =
            "^\\s*driver.findElementById\\(\".*:id/.*\"\\).sendKeys\\(.*\\);$";
    public static final String SEND_KEY_EVENT =
            "^\\s*driver.sendKeyEvent(.*);$";
}
```

图 4.5: 测试代码匹配正则表达式

如图 4.6所示，为建立事件模型与测试脚本映射的具体代码，该工作由MappingService.java完成。首先通过getVertices方法从数据库中读取当前版本应用的事件流图模型数据，然后使用Java语言原生的BufferedReader读取测试脚本文件，按行分析代码语句，当然这里要过滤到Java语言的基本代码，如类、方法的申明和注释等，由isJavaBaseCode方法完成这部分工作。提取出测试脚本中的测试代码，对其进行归类，按照图 4.5罗列的正则表达式来判断其定位GUI控件的方式，提取出用于控件定位的参数，再从事件流图模型的点集中检索出与之对应的节点，从而为事件流图模型节点和对应的测试代码建立映射关系，这里对于无法检索到与之对应节点的测试代码，我们也为其建立映射关系，但是该测试代码对应的事件节点为空，表示测试代码对应的GUI控件并不存在与本版本的应用中，codeToMapping方法封装即是对于不同定位方式测试代码的处理逻辑。

```
public void start(String path, String version, Long appId){
        List<Vertice> vertices = verticeMapper.getVertices(version);
        File script = new File( DirConstants.SCRIPT_DIR + appId + "_" + version + "/"
+ path);
        try{
            FileInputStream inputStream = new FileInputStream(script);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
            int line = 1;
            String currentActivity = "";
            for (String res; (res = reader.readLine()) != null; line++) {
                if (StringUtils.isEmpty(res) || isJavaBaseCode(res)){
                    continue;
                } else if (res.matches(CodePattern.WAIT_FOR_ACTIVITY)){
                    String[] s = res.split("\"");
                    currentActivity = s[1];
                    continue;
                }else {
                    codeToMapping(res, line, version, vertices, false, currentActivity);
                }
            }
        } catch (FileNotFoundException e){
            // … 处理异常
        } catch (IOException e){
            // … 处理异常
        }
    }
```

图 4.6: 建立模型脚本映射代码

4.2.3　映射准确度评估

本文对于映射准确度的评估通过实验人工审核统计，如图 4.7所示，为移动应用咕咚翻译1.8.3版本的模型与上一版本1.8.0测试脚本的映射数据。图中数据记录了1.8.0版本测试脚本中的所有测试代码、代码的行号以及它们所对应的事件流图模型节点的编号。经过统计，1.8.0 版本的测试脚总计357行，其中GUI控件定位执行事件的测试代码总计266行，其中265行测试代码存在与其对应的事件流图模型节点，只有1 行映射为空。具体的实验过程和结果数据将在下一章展示。

id	vid	code	line_code	version
153	73	driver.findElementByXPath("//android.widget.Button[@text='已了解']").click();	39	1.8.0
154	74	driver.findElementByXPath("//android.widget.Button[@text='确定']").click();	40	1.8.0
155	57	driver.findElementById("android:id/input").click();	41	1.8.0
156	57	driver.findElementById("android:id/input").sendKeys("test");	42	1.8.0
157	59	driver.findElementById("name.gudong.translate:id/tv_clear").click();	43	1.8.0
158	57	driver.findElementById("android:id/input").sendKeys("test");	44	1.8.0
159	58	driver.findElementById("name.gudong.translate:id/bt_translate").click();	45	1.8.0
160	60	driver.findElementById("name.gudong.translate:id/sp_translate_way").click();	46	1.8.0
161	61	driver.findElementByXPath("//android.widget.CheckedTextView[@text='百度']").click();	47	1.8.0
162	60	driver.findElementById("name.gudong.translate:id/sp_translate_way").click();	48	1.8.0
163	75	driver.findElementByXPath("//android.widget.CheckedTextView[@text='谷歌']").click();	49	1.8.0
164	60	driver.findElementById("name.gudong.translate:id/sp_translate_way").click();	50	1.8.0
165	76	driver.findElementByXPath("//android.widget.CheckedTextView[@text='金山']").click();	51	1.8.0
166	60	driver.findElementById("name.gudong.translate:id/sp_translate_way").click();	52	1.8.0
167	77	driver.findElementByXPath("//android.widget.CheckedTextView[@text='有道']").click();	53	1.8.0
168	78	driver.findElementById("name.gudong.translate:id/iv_favorite").click();	54	1.8.0
169	79	driver.findElementById("name.gudong.translate:id/iv_sound").click();	55	1.8.0
170	80	driver.findElementById("name.gudong.translate:id/iv_paste").click();	56	1.8.0
171	63	driver.findElementByXPath("//android.widget.ImageView[@content-desc='更多选项']").click();	57	1.8.0
172	81	driver.findElementByXPath("//android.widget.TextView[@text='历史记录']").click();	58	1.8.0
173	71	driver.findElementByXPath("//android.widget.ImageButton[@content-desc='转到上一层级']").click();	60	1.8.0
174	63	driver.findElementByXPath("//android.widget.ImageView[@content-desc='更多选项']").click();	61	1.8.0
175	82	driver.findElementByXPath("//android.widget.TextView[@text='支持作者']").click();	62	1.8.0
176	83	driver.findElementByXPath("//android.widget.Button[@text='关闭']").click();	64	1.8.0
177	63	driver.findElementByXPath("//android.widget.ImageView[@content-desc='更多选项']").click();	65	1.8.0
178	84	driver.findElementByXPath("//android.widget.TextView[@text='去评分']").click();	66	1.8.0
179	63	driver.findElementByXPath("//android.widget.ImageView[@content-desc='更多选项']").click();	67	1.8.0
180	85	driver.findElementByXPath("//android.widget.TextView[@text='关于(1.8.0)']").click();	68	1.8.0
181	71	driver.findElementByXPath("//android.widget.ImageButton[@content-desc='转到上一层级']").click();	70	1.8.0
182	63	driver.findElementByXPath("//android.widget.ImageView[@content-desc='更多选项']").click();	71	1.8.0
183	86	driver.findElementByXPath("//android.widget.TextView[@text='设置']").click();	72	1.8.0
184	87	driver.findElementByXPath("//android.widget.Button[@text='知道了']").click();	74	1.8.0
185	88	driver.findElementByXPath("//android.widget.TextView[@text='开启划词翻译']").click();	75	1.8.0
186	89	driver.findElementByXPath("//android.widget.TextView[@text='开启自动发音']").click();	76	1.8.0
187	90	driver.findElementByXPath("//android.widget.TextView[@text='打开 App 自动翻译粘贴板单词']").click()	77	1.8.0
188	91	driver.findElementByXPath("//android.widget.TextView[@text='开启定时单词提醒']").click();	78	1.8.0
189	92	driver.findElementByXPath("//android.widget.TextView[@text='提示显示时间']").click();	79	1.8.0
190	93	driver.findElementByXPath("//android.widget.CheckedTextView[@text='3秒钟']").click();	80	1.8.0

图 4.7: 模型脚本映射结果部分数据展示

4.3　脚本修复模块

4.3.1　脚本修复模块概述

本系采用人机协同的修复方式，包含自动化修复的工作也提供人工修复的功能，本模块实现的是测试脚本自动化修复。有了模型生成模块和映射生成模

块的工作基础，本模块就可以开始进行脚本的修复工作。从图 4.7中，我们可以看出映射生成模块所建立的映射关系一般分为两种，第一种存在与测试代码对应的事件节点，则说明该测试代码对应的事件在新版本应用上是合法的，也即是该事件涉及的GUI控件在新版本应用上是存在的。第二种测试代码对应的事件节点为空，则说明该测试代码对应的事件在新版本应用上是不合法的，也即是新版本应用上不存在该事件涉及的GUI控件，或者该事件涉及的GUI控件的定位参数发生了变化，使得本该映射上的测试代码映射为空。本模块结合了事件流图模型和模型映射关系实现了基于图最短径的修复算法 [38]。

4.3.2　脚本修复算法实现

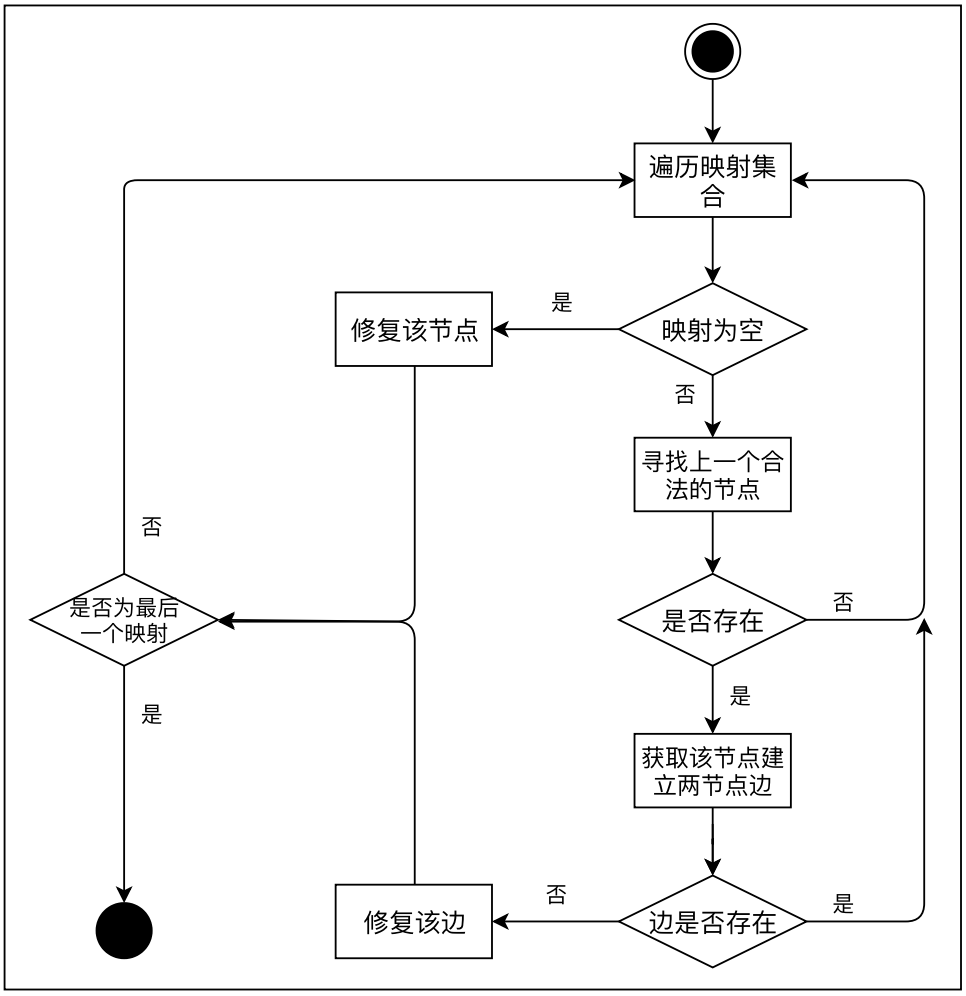

图 4.8: 修复算法流程图

本模块根据GUI的改变将修复分为四个类别：GUI元素增加和删除，元素间关系的删除，元素属性的变化，构建自动修复算法。修复过程如图 4.8 所示。

```
public void autoRepair(Long appId, String targetVersion, String fileName){
        // … 获取指定版本的事件流图模型数据
        graphUtil = new GraphUtil(vertices, edges); // 初始化图
        // … 获取所有测试代码行号
        try {
            int index = 0;
            for (int i = 0; i< oldCodes.size(); i++){
                if (lines.indexOf(i + 1) == -1){
                    // … 不是测试代码直接输出
                }else {
                    // … 判断映射是否已删除
                    if (mapping.getVid() == null) {
                        // … 修复事件模型节点
                    } else {
                        if (index > 0) {
                            // … 向前找上一个未被删除的代码映射
                            if (preIndex > -1) {
                                // … 获取该映射并建立检查边
                                if (from != to && !edges.contains(e)) {
                                    // … 修复事件流图模型边
                                } else {
                                // … 如果边存在，或者两句代码映射为同一个事件，直接输出
                                }
                            } else {
                        // … 如果向前未能找到上一个未被删除的代码映射，直接输出
                            }
                        } else {
                            // … 如果是第一句直接输出
                        }
                    }
                    index ++;
                }
            }
        } // … 处理异常
}
```

图 4.9: 修复算法部分代码

如图 4.9所示，为修复算法的部分代码。首先遍历所有的映射，检查每一个映射对应的节点是否存在于模型中，如果不存在则修复该点，如果存在则验证该点与上一个合法的映射关系对应的点所形成的关系是否存在于事件流图模型的边集中，如果该关系不存在则修复此边。图模型中边的修复，通过弗洛伊德最短路径算法寻找合法的路径来替换的方式。而结点的修复，首先删除此不存在的映射，寻找上一个合法的映射对应的点，以及下一个映射不为空的点，如果两个节点对应的元素相同，则需要特殊判断，如果该点为父组件，则需要删

除重复的测试代码。这样设计是为了避免由于父子组件的GUI显示关系，重复点击父组件可能会导致子组件的隐藏，从而关于子组件的测试代码运行失败。如果两个映射对应的点不相同，则同样寻找最短路径来修复此边。当整个遍历过程结束，也就确保了测试脚本所执行的路径在新版本的应用中全部合法，而测试脚本也就能成功执行结束，能够得到复用 [39]。

4.3.3　修复效果评估

本文的效果评估主要从两个方面来进行，分别是代码层面和界面层面。总结为两个指标：SCP（Screen Coverage Presentation）和TAP（Test Action Presentation） [11]。SCP即旧版本脚本覆盖的界面或事件维护后依然能被覆盖的比例。TAP即废弃的测试代码语句恢复可用的比例，即本次维护使得多少废弃的测试代码语句恢复可用。具体的计算方式如下：

$$SCP = \frac{|S'v \cap Sc|}{|Sv \cap Sc|} \tag{4.1}$$

Sc表示应用总界面数，Sv表示修复之前测试脚本覆盖的界面数，$S'v$表示修复之后测试脚本覆盖的界面数。

$$TAP = \frac{|(A'e - Ae) \cap At|}{|At - Ae|} \tag{4.2}$$

At表示测试脚本总代码数，Ae表示修复之前测试脚本能够执行的代码数，$A'e$表示修复之后测试脚本能够执行的代码数。

具体的实验数据会在下一章中给出。

4.4　人机协同模块

4.4.1　人机协同模块概述

正如前文所述，自动化修复的效果很大程度上取决于事件流图模型的准确度，也即是取决于模型生成模块所使用的自动化遍历工具的效果。鉴于目前安卓应用碎片化严重的情况，自动化遍历工具对于应用的遍历效果受到应用本身的兼容性、移动设备的影响非常大，所得到遍历结果无法达到理想中最好的效果。而且，即使工具能够遍历到所有的安卓GUI控件，工具采用的遍历算法并非能够覆盖到所有的GUI路径。从另一个角度来讲，一份测试脚本的编写往往蕴含了测试人员的智慧，包含了测试人员对于测试步骤的设计思想，所以测试路径并不一定能够存在于自动化遍历工具的输出结果中。

因此，为了解决这个问题并提高修复成功率，本模块提供了人机协同的交互方式，将对应版本应用的事件流图模型以可编辑的有向图的形式，通过Web UI展现给测试用户，直观地表示安卓应用GUI的结构，同时我们也提供不同版本事件流图对比的功能和事件流图模型的编辑功能，让测试用户人工地修正事件流图模型的结构，从而提高自动化遍历工具结果所生成的事件流图模型的准确率。本模块将人工审查测试脚本转变为人工编辑更为直观的图模型，将人工修改测试脚本转变为将图模型的修改映射到测试脚本的修复上，来提高测试脚本维护的效率。

4.4.2　人机协同交互实现

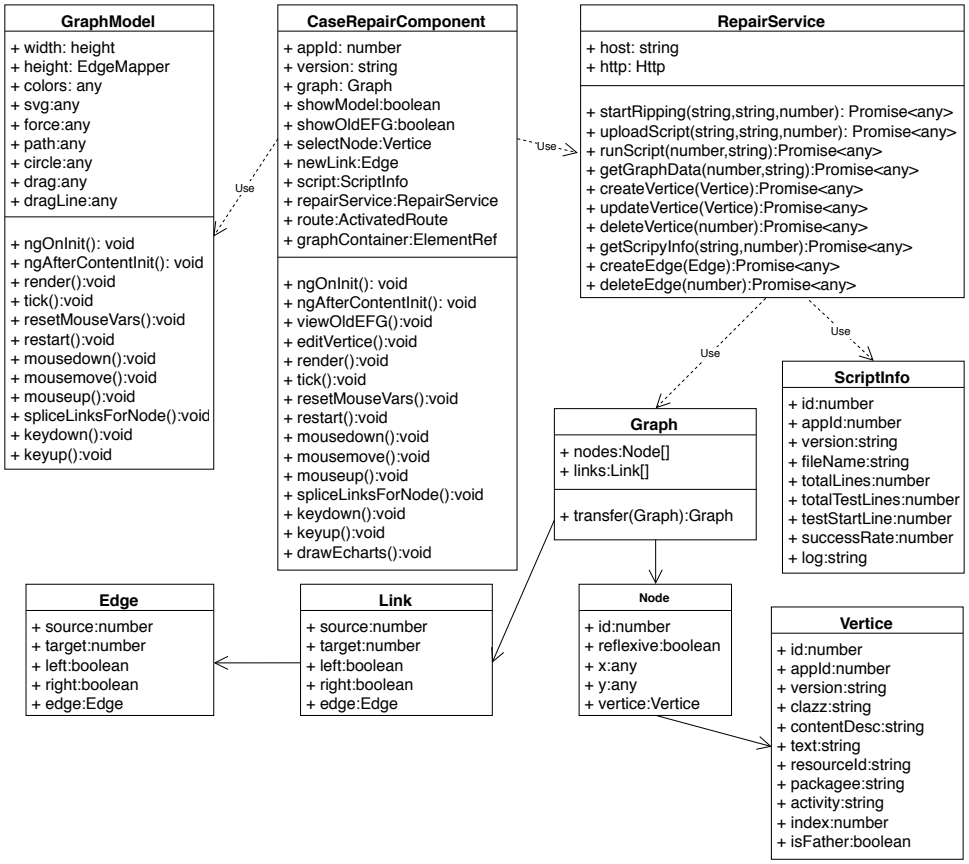

图 4.10: 人机协同交互设计类图

如图4.10所示，为人机协同交互模块的设计类图。GraphRepairComponent为修复界面的控制器，负责事件模型的渲染过程和界面各种交互的逻辑实现。RepairService为前者注入需要的服务，实现了前后端的业务交互，包括模型和脚本数据的读写。Graph、ScriptInfo等类定义了前后端数据交互的结构。

```
// 初始化D3的SVG
const width = 1280;
const height = 1000;
const colors = d3.scaleOrdinal(d3.schemeCategory10);
const svg = d3.select('body')
        .append('svg')
        .attr('oncontextmenu', 'return false;')
        .attr('width', width)
        .attr('height', height);
//初始化D3的力导向图的模拟器
const force = d3.forceSimulation()
        .force('link', d3.forceLink().id((d) => d.id).distance(150))
        .force('charge', d3.forceManyBody().strength(-500))
        .force('x', d3.forceX(width / 2))
        .force('y', d3.forceY(height / 2))
        .on('tick', tick);
// … 设置拖拽事件
// … 绘制力导向图的边与箭头
this.svg.on('mousedown', (dataItem, value, source) => this.mousedown(dataItem,
value, source))
        .on('mousemove', (dataItem) => this.mousemove(dataItem))
        .on('mouseup', (dataItem) => this.mouseup(dataItem)); // 绑定鼠标监听事件
// … 绑定键盘监听事件
d3.select(window)
    on('keydown', this.keydown.bind(this))
    on('keyup', this.keyup.bind(this));
```

图 4.11: 使用D3.js绘制事件流图模型代码

本模块使用D3.js的力导向图（Force Directed Graph）来展示应用的事件流图模型，使用可编辑有向图（Directed Graph Editor）实现人机协同的交互，如图 4.11 所示，为人机协同交互模块使用D3.js初始化事件流图的部分代码。这里首先定义一个SVG画布，设置其长宽和颜色序列，d3.schemeCategory10为RGB十六进制表示的二十种颜色数组，然后创建一个力学模拟器forceSimulation，为整个图形设置一个位于图形中心的居中力，之后设置图形的拖拽力导向图的边与方向箭头，设置力学模拟器内部计时器的结束事件，用于更新力导向图布局。而最后的restart()方法定义了力导向图更新的具体过程，包括为力学模拟器绑定节点数据和连接线数据、为节点和连接线设置index等五个数据等过程。这里对于图中节点的展示，本模块优先使用事件节点的text和content-desc属性，因为用户在应用中可以直接看到GUI控件的这两个属性值。如果事件节点这两个属性值为空，则采用resource-id 属性或class +index属性显示，以达到直观地展示应用事件流图的效果。

事件流图模型初始化完毕后，还需要为编辑图模型设置监听事件，具体代码如图 4.12所示，分别为事件流图模型设置了鼠标点击事件、鼠标移动事件、鼠标点击释放事件、键盘各键位点击事件、键盘各键位点击释放事件等。这里键盘鼠标点击事件对应着用户添加新的事件流图节点行为，鼠标点击释放事件对应着用户添加新的事件流图边行为。最为复杂和重要的是键盘点击事件，键盘Del键的点击事件对应着用户删除事件流图节点或边行为，键盘R键、B键和L键点击事件对应着用户修改事件流图边的方向事件。

```
function mousedown() {
  // … 鼠标点击事件
  // … 添加新的节点
  restart();
}
function mousemove() {
  // … 鼠标移动事件
}
function mouseup() {
  // … 鼠标点击松开事件
}
function keydown() {
  d3.event.preventDefault();
  if (lastKeyDown !== -1) return;
  lastKeyDown = d3.event.keyCode;
  if (d3.event.keyCode === 17) {
    circle.call(drag);
    svg.classed('ctrl', true);
  }
  if (!selectedNode && !selectedLink) return;
  switch (d3.event.keyCode) {
    // … 处理Del等键的事件
  }
}
function keyup() {
  lastKeyDown = -1;
  if (d3.event.keyCode === 17) {
    circle.on('.drag', null);
    svg.classed('ctrl', false);
  }
}
```

图 4.12: 设置事件流图的监听事件代码

4.4.3　人机协同修复实现

本模块主要负责的业务由三部分组成。第一部分负责接收并存储测试用户编辑事件流图模型的数据，包括节点更新和新增边等数据。第二部分负责根据事件流图模型的编辑数据生成新的测试脚本，对于不同类型的事件流图模型变化，本模块对应的处理逻辑也不同。第三部分负责调用移动应用自动化测试服务运行新生成的测试脚本，然后获取运行结果数据，再经过计算处理后将成功率和日志等数据反馈给测试用户，这三部分业务不断循环运行构成了完整的人机协同修复的业务流程。

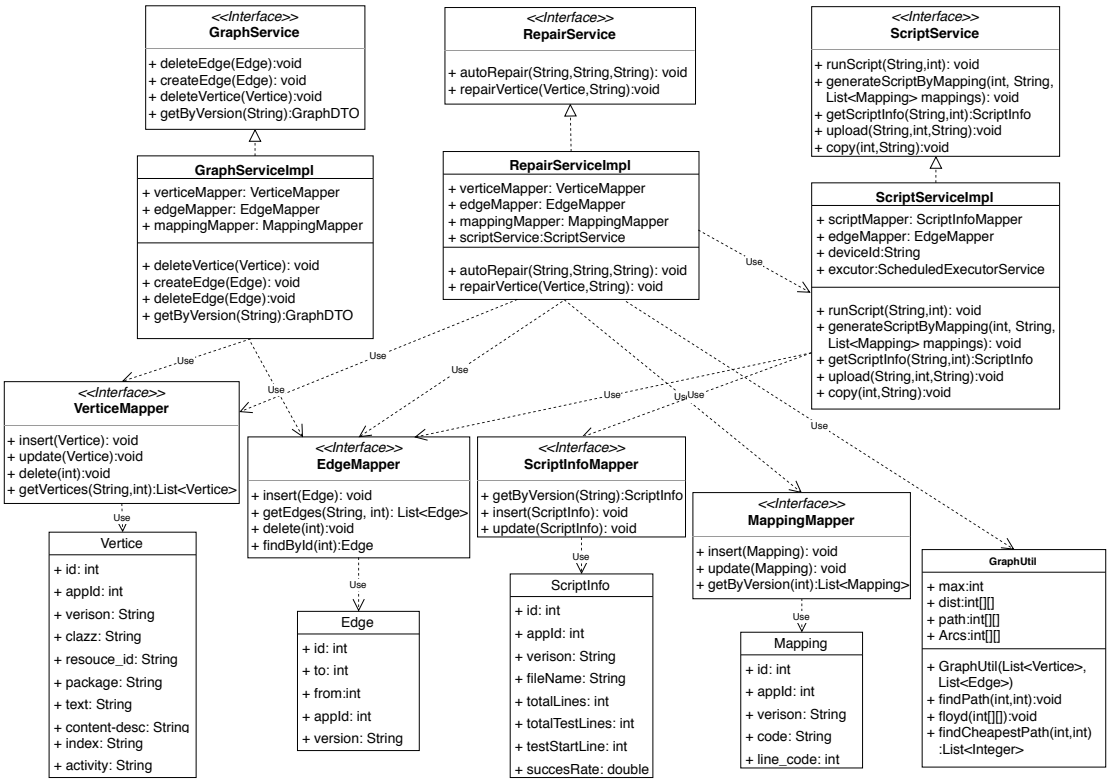

图 4.13: 人机协同修复设计类图

如图 4.13所示，为人机协同修复的设计类图。服务层主要由GraphService、ScriptService和RepairService组成，分别封装了事件流图模型数据的读写、脚本运行结果数据的读写和脚本修复服务的业务逻辑，同时RepairService会调用ScriptService触发测试脚本的运行业务。数据访问层（DAO）由VerticeMapper、EdgeMapper、MappingMapper、ScriptInfoMapper等类组成，提供数据库读写业务。GraphUtil 类封装了图的最短径算法——弗洛伊德算法的实现逻辑，提供获取事件流图模型中两个节点之间最短路径的方法。持久化对象（Persistent

Object，PO）由Veritce、Edge、Mapping、ScriptInfo 等类组成，封装了事件流图模型节点、事件流图模型边、模型脚本映射、测试脚本信息等数据结构。

如图 4.14所示，为人工修复事件节点的部分代码。首先根据前端所传递的节点id通过数据访问层verticeMapper和mappingMapper获取旧的节点数据和该节点对应的模型映射数据。然后根据脚本映射生成模块定义好的正则表达式判断该节点对应的模型映射中的测试代码定位GUI控件的方式，再根据不同的定位方式分别进行参数替换，生成新的测试代码。然后更新对应节点的数据和该节点对应的模型脚本映射数据，并调用修复算法进行测试脚本修复，最后生成新的测试脚本。

```
public void repairVertice(Vertice vertice, String apkName){
    Vertice oldVertice = verticeMapper.findById(vertice.getId());
    verticeMapper.update();
    List<Mapping> mappings = mappingMapper.getByVertice(vertice.getId());
    for (Mapping mapping: mappings){
        String code = mapping.getCode();
        String newCode = null;
        // 判断测试代码的GUI控件定位方式
        if (code.matches(CodePattern.SENDKEYS_BY_ID)
&& !oldVertice.getResourceId().equals(vertice.getResourceId())){
            // … 处理使用resource_id定位
        } else if (code.matches(CodePattern.POSTION_BY_ID)
&& !oldVertice.getResourceId().equals(vertice.getResourceId())){
            // … 处理使用resource_id定位
        } else if (code.matches(CodePattern.POSTION_BY_TEXT)
&& !oldVertice.getText().equals(vertice.getText())){
            // … 处理使用text定位
        } else if (code.matches(CodePattern.POSTION_BY_CONTENTDESC)
&& !oldVertice.getContentDesc().equals(oldVertice.getContentDesc())){
            // … 处理使用content-desc定位
        } else if (code.matches(CodePattern.POSTION_BY_INDEX) &&
oldVertice.getIndex() != vertice.getIndex()){
            // … 处理使用index定位
        }
        // … 更新数据库
    }
    // … 生成新的脚本
}
```

图 4.14: 人工修复事件节点部分代码

4.5　本章小结

本章详细介绍了安卓GUI测试脚本修复系统模型生成模块、映射生成模块、脚本修复模块、人机协同模块这四个核心模块的详细设计和实现，主要包括各个模块的设计类图、部分实现代码以及修复算法的流程图、设计思路等，并展示了部分系统的界面截图。本章对每个模块的效果设计评估方案，通过模拟实验的部分数据展示验证了各个模块设计和实现方案的合理性，这四个模块构建了整个安卓GUI测试脚本修复系统的核心工作流。

第五章　安卓GUI测试脚本修复系统测试和实验设计

为了验证安卓GUI测试脚本修复系统的可用性，本文进行了系统测试和修复实验的设计。本文设计了多个功能性测试用例验证系统业务流程的可用性。本文使用7款安卓应用的不同版本来进行修复实验，通过修复过程中事件流图模型的准确度、模型脚本映射的准确度、修复后测试脚本的成功率、修复结果的TAP和SCP以及修复效率来进行验证。我们考虑通过与AndroidRipper [40]工具所生成的图模型对比来验证事件流图模型的准确度，但是AndroidRipper对于安卓版本的支持最多只能到7.0版本，对于近期的安卓应用和设置支持度不好。在实验中，我们希望能够验证以下几个问题。

（1）问题一：本文的自动化遍历工具生成的事件流图模型准确度如何？

（2）问题二：本文的系统所有功能是否可用？系统的交互和运行是否符合预期的设计？

（3）问题三：本文的系统修复效果如何？修复后测试脚本运行的成功率如何？

5.1　测试准备

5.1.1　测试环境

如表 5.1所示，为本系统的测试开发环境，包括所涉及硬件资源和软件资源的版本。本文的系统前后端程序和数据库等部署和运行在真实的Linux 系统设备上，并连接真实的安卓设备进行测试。

表 5.1: 测试环境配置表

设备与软件	备注
服务器	内存16GB，带宽10MB
部署系统	Ubuntu 16.04
数据库	MySQL 5.7
开发系统	MacOS 10.13.5
反向代理	Nginx 1.12.2
Java编译环境	JDK 1.8
后端打包工具	Maven 3.5.2
前端运行环境	Nodejs 8.9.0
手机设备操作系统	Android 7.0

5.1.2　待测应用

本文采用真实的安卓应用产品进行测试，从Github上选取了7个开源的安卓应用，并为每个应用挑选了其相邻的两个版本。在进行应用挑选时，本文充分考虑了应用的多样性，挑选了尽可能多不同类别的应用。为了验证安卓GUI测试脚本修复系统的可行性，本文确保每个应用的所选版本存在GUI变更，以咕咚翻译为例，如图 5.1所示为咕咚翻译两版本的GUI变化。如表 5.2所示，为所选应用的编号、名称、类型、版本号等数据。

表 5.2: 待测应用信息表

编号	名称	类型	版本一	版本二
A1	咕咚翻译	工具	1.8.0	1.8.3
A2	Leafpic	工具	0.3.1	0.3.5
A3	Bilibili	娱乐	2.3.0	2.3.1
A4	IThouse	资讯	1.0.0	1.0.1
A5	GnuCash	理财	2.3.0	2.4.0
A6	Ve2x	社交	1.2.5	1.2.10
A7	AntenaPod	音乐	1.6.5	1.7.0

图 5.1: 咕咚翻译1.8.0版本和1.8.3版本

5.2　测试过程和实验设计

5.2.1　功能测试

本节根据第三章中描述的系统功能，为安卓GUI测试脚本修复系统设计了功能性测试用例。表 5.3等为主要测试用例对应的用例描述表。

表 5.3: 创建待测应用测试用例

用例编号	UC1
用例名称	创建待测应用
用例描述	用户在系统上创建待测的安卓应用，系统存储应用的apk文件，并调用自动化遍历工具对该应用进行扫描，为其生成事件流图模型
测试步骤	1.登录系统，进入应用版本列表页面 2.选择上传新版本 3.选择对应的安卓应用文件并上传 4.提交待测应用
预期结果	1.应用版本列表显示上传的新版本应用信息 2.可以下载对应的应用文件 3.在测试脚本修复界面可以看到该应用的事件流图模型

表 5.3描述的是创建待测应用测试用例的执行过程。本文设计了边界测试，创建重复版本的应用，系统应该提示创建失败，已存在该版本应用。

表 5.4: 拷贝测试脚本测试用例

用例编号	UC2
用例名称	拷贝测试脚本
用例描述	用户在系统中为新版本的安卓应用，拷贝旧版本的测试用例用于新版本应用的测试
测试步骤	1.登录系统，进入应用版本列表页面 2.选择拷贝测试用例并选择需要拷贝对应的应用版本 3.选择需要拷贝的测试用例 4.确认拷贝
预期结果	1.新版本应用的测试用例列表显示拷贝的测试用例信息 2.可以下载对应的测试脚本文件

表 5.4描述的是拷贝测试用例的执行过程。系统对于已经拷贝过的测试用例，在待拷贝用例列表应该不再显示。

表 5.5: 添加事件模型节点测试用例

用例编号	UC3
用例名称	添加事件模型节点
用例描述	用户在系统中为待测应用的事件流图模型添加新的节点
测试步骤	1.登录系统，进入测试脚本修复页面 2.点击事件流图模型空白处 3.输入该节点对应的信息 4.提交节点数据
预期结果	1.系统提示添加成功 2.事件流图中出现新添加的节点，用户可以查看、编辑该节点的信息件

表 5.5描述的是添加事件模型节点测试用例的执行过程，表 5.6描述的是添加事件模型边测试用例的执行过程。新添加的节点和边，页面刷新后应该正常显示，并且可以编辑、删除。

表 5.6: 添加事件模型边测试用例

用例编号	UC4
用例名称	添加事件模型边
用例描述	用户在系统中为待测应用的事件流图模型添加新的边
测试步骤	1.登录系统，进入测试脚本修复页面 2.选择节点作为边起点 3.连接终点节点
预期结果	1.系统提示添加成功 2.事件流图中正确出现新添加的边，用户可以删除该边 3.数据库中正确增加对应边的数据

表 5.7: 删除事件模型节点测试用例

用例编号	UC5
用例名称	删除事件模型节点
用例描述	用户在系统中为待测应用的事件流图模型删除原有的节点
测试步骤	1.登录系统，进入测试脚本修复页面 2.选择要删除的节点 3.点击删除
预期结果	1.系统提示删除 2.事件流图中正确删除指定的边，用户可以再次添加该边 3.数据库中正确删除对应的边数据

表 5.7描述的是删除事件模型边测试用例的执行过程。删除的节点，页面刷新后应该不再显示，并且可以再次添加。

表 5.8: 运行测试脚本测试用例

用例编号	UC6
用例名称	运行测试脚本
用例描述	用户在系统中选择修复运行测试脚本，系统进行测试脚本自动化修复，修复完毕生成新的测试脚本后调用服务运行测试脚本，用户在系统中能够查看测试脚本运行结果
测试步骤	1.登录系统，进入运行结果界面 2.点击修复运行按钮
预期结果	1.系统提示测试脚本运行中 2.运行结果后系统显示测试脚本的运行结果，包括运行日志，测试脚本的成功率

表 5.8描述的是运行测试脚本测试用例的执行过程。脚本运行结束后，页面应该由运行中转变为显示成功率和日志等信息。

5.2.2　实验设计

本文的实验步骤如下：

（1）为每个应用版本一准备测试脚本并确保都是能够成功运行的脚本

（2）使用本系统为每个应用的两个版本建立事件流图模型

（3）统计事件流图模型的准确度

（4）使用本系统为每个应用版本二进行测试脚本修复

（5）统计测试脚本修复结果

对于步骤一，本文采用移动应用测试平台现有基于Appium测试框架编写的测试脚本，考虑到移动应用测试平台的测试脚本来源更贴近真实的生产场景。这里我们通过移动应用自动化测试服务批量运行测试脚本并确保选取其中成功的那些脚本，去除由于设备兼容性而失败的部分脚本。对于测试脚本不足的应用，我们采用人工编写脚本的方式进行补充。如表 5.9为所用测试脚本的数量。

表 5.9: 各应用测试脚本数量

应用编号	脚本数量
A1	15
A2	15
A3	9
A4	9
A5	12
A6	16
A7	20

5.3　测试结果

5.3.1　测试执行结果

如表 5.10所示，为上述功能测试用例的执行结果表，所有测试用例执行均通过。测试用例执行的界面截图如图 5.2- 5.6所示。

表 5.10: 功能测试用例执行结果表

用例编号	对系统用例编号	执行结果
UC1	C1	通过
UC2	C3	通过
UC3	C6	通过
UC4	C6	通过
UC5	C6	通过
UC6	C5	通过

如图 5.2所示，为创建待测应用界面截图，该界面展示的是目标应用的版本列表和所选版本测试用例列表，测试用户点击上传新版本按钮，可以创建待测应用，点击选择应用按钮选择应用文件进行上传，如果上传已有版本的应用，系统会提示已存在相同版本的应用。

图 5.2: 创建待测应用界面截图

如图 5.3所示，为拷贝测试用例界面截图，该界面展示的是目标应用历史版本对应的测试脚本列表，用户可以选择指定历史版本的指定测脚本将其拷贝到当前版本应用上用于测试，这里系统会过滤已经拷贝过的测试用例，使其不会再出现在列表中。

图 5.3: 拷贝测试用例界面截图

如图 5.4所示，为人机协同修复中事件模型界面的截图，该界面展示的即为待测应用的事件流图模型，测试用户在本界面可以看到目标应用的所有GUI 事件和逻辑关系，并对其进行编辑，通过修改事件节点之间的关系或者修改事件节点的相关信息来进行人工修复。

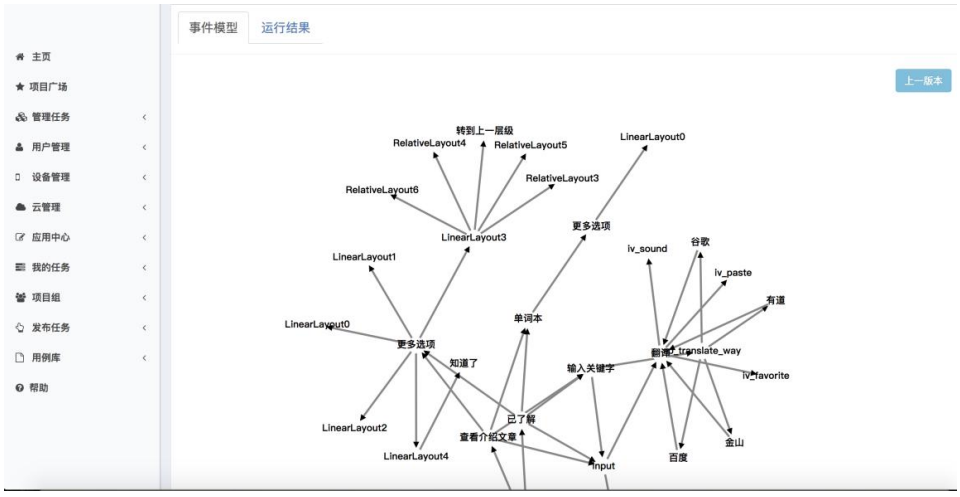

图 5.4: 应用事件模型界面截图

如图 5.5所示，为人机协同修复中事件模型节点信息界面的截图，该界面展示的是测试用户所选中的GUI事件节点对应的各种信息，包括类型、内容、标识等一系列测试脚本用于定位GUI控件的数据。

图 5.5: 事件模型节点信息界面截图

如图 5.6所示，为人机协同修复中脚本运行结果界面的截图，这里展示该测试脚本修复运行成功率，脚本本身的信息包括代码行数等，以及运行测试脚本时收集到的相关日志输出。

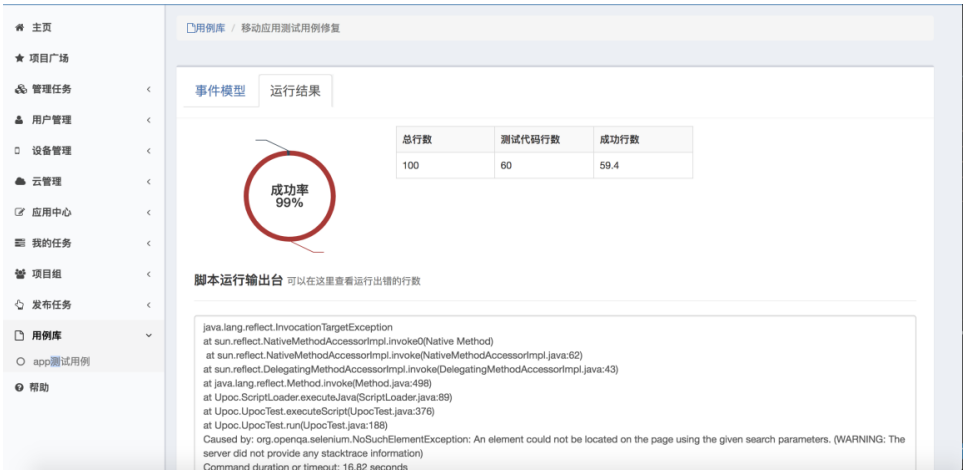

图 5.6: 脚本运行结果界面截图

5.3.2　实验结果

我们结合Uiautomatorviewer工具进行人工统计了各应用的GUI控件，与自动化遍历工具生成的事件流图模型数据进行对比，表 5.11是统计的准确度结果，精确度是由自动化工具所扫描生成的事件流图模型节点数与人工统计GUI控件数做除运算所得。由于安卓应用界面的复杂性，人工统计图模型的边即GUI控件的逻辑关系工作量非常巨大，不太现实，这里就不展示关于边的具体数据。

由表中数据我们可以发现自动化遍历工具能够遍历出的事件节点无法达到百分之百的准确度，这一问题在本文中通过人机协同的方式的得到解决，但后续的工作还需要进一步改进自动化工具的遍历效果，以求建立更准确的事件流图模型从而进一步降低维护成本，提高修复效率。

表 5.11: 各应用事件流图模型的准确度

编号	模型准确度
A1	88.37%(38/43)
A2	88.5%(54/61)
A3	75.3%(101/134)
A4	82.63%(119/144)
A5	84.78%(39/46)
A6	76.41%(81/106)
A7	78.18%(43/55)

本文统计修复前后各应用测试脚本的成功率，并计算了SCP和TAP两个参数 [11]，这里的成功率是每个应用所有测试脚本中成功运行通过的测试代码行数除以测试代码总数的结果取均值所得，而SCP和TAP上文已经做了详细的解释，这里就不再具体解释。

表 5.12: 各应用测试脚本修复结果

应用编号	修复前成功率	自动化修复成功率	人机协同修复成功率	SCP	TAP
A1	15.9%	52.3%	90.4%	1(5/5)	0.78(43/55)
A2	9.3%	48.4%	89.2%	0.91(10/11)	0.79(54/69)
A3	33.3%	35.1%	70.1%	1(7/7)	0.42(72/154)
A4	19.2%	38.9%	82.4%	0.85(11/13)	0.63(122/195)
A5	21.6%	51.5%	80.5%	0.88(7/8)	0.58(29/56)
A6	16.3%	40.6%	73.2%	1(15/15)	0.57(75/131)
A7	5.4%	35.9%	75.4%	1(12/12)	0.68(60/88)

如表 5.12所示，为各应用的测试脚本修复效果数据。从表中的数据我们可以看出，整体的测试脚本成功率在修复之后得到大大的提高。基于工具建立的事件流图模型的自动化修复效果不是很理想，成功率略低原因有几种，第一种为目前自动化遍历工具采用深度遍历策略，所得到的GUI事件路径不是很全面。第二种为应用的GUI界面变化使得整个界面布局超过了移动设备的屏幕大小，这里需要向测试脚本添加滑屏事件，目前的自动化工具尚不能扫描出这种情况，需要在后续的工作中进一步改进工具。本文引入了人机协同修复的方式，提供对应用的事件流图模型的人工补全功能，从结果上来看，大大提高了测试脚本的修复成功率。

5.4　本章小结

本章进行了安卓GUI测试脚本修复系统的系统测试和实验设计。对功能测试和实验结果总结与分析。本文详细描述了系统测试的环境和设计、对比实验的步骤，展示了测试结果和实验结果的数据。本文通过对不同应用多版本的实验，验证了修复工作的效果和系统的可用性，分析了修复效果和修复效率上的优劣。

第六章　总结和展望

6.1　总结

随着智能手机的不断普及和移动互联网技术的蓬勃发展，各行各业都想方设法地把自己业务或产品和移动互联网结合起来，由此使得安卓应用的数量不断增长，安卓应用的迭代速度也不断加快，应用新的版本也不断地发布，尤其是对于很多处于上升期的移动应用产品，几乎达到一周一个小版本。为了应对如此之快的版本变更速度，也为了应用产品的质量保障，移动应用自动化测试技术也得到广泛应用，其中使用最广泛的是安卓GUI自动化测试框架Appium。而版本的更替往往会带来GUI的变化，如GUI控件的增删改等。而这会直接导致安卓应用界面的事件流的变化，从而使得先前版本的测试脚本无法直接复用，而由于企业测试资源的匮乏，人工维护测试脚本的成本非常大。安卓GUI测试脚本修复系统为此问题提供了解决方案。

本文介绍安卓GUI测试脚本修复系统的项目背景、业务背景和国内外相关研究现状，详细介绍了本文的研究工作。本文阐述安卓GUI测试脚本修复系统的开发所使用到的技术，本系统使用Spring Boot和Mybatis来构建服务端程序，使用Angular和D3.js来构建前端的程序，使用Appium和Uiautomator来构建自动化遍历工具和测试脚本运行服务，使用Dubbo和Zookeeper来进行服务的远程调用。本文对安卓GUI测试脚本修复系统的需求分析、架构设计和模块划分做了详细的阐述，接着详细介绍了各个系统模块的详细设计和具体实现。

本文构建了一个基于事件模型和人机协同的安卓GUI测试脚本为解决安卓版本变化而测试脚本维护成本过高的问题提供了一种解决思路。并通过实验设计和数据论证了系统的可用性。

6.2　展望

目前安卓GUI测试脚本修复系统已经开发完成，主要的脚本修复相关功能模块都已完成，并且经过了一定的实验，初步实现了一个工业级的应用，同时还存在以下两个问题，有待后续的工作进行改善。

（1）本系统目前研究的测试脚本均是基于Appium测试框架所编写的，虽然Appium在工业界的应用非常广泛，但还存在不少其他的测试框架如Espresso、Robotium等，本系统目前对相关功能进行了接口的封装，后续的工作会为不同

的测试框架进行接口实现，将现有测试脚本修复技术移植到其余其他测试框架技术上从而丰富修复系统的功能。

（2）基于事件流模型的测试脚本修复方式，修复效果在很大程度上取决于事件流模型的完整程度，目前本系统采用自研的自动化遍历工具进行应用的扫描从而建立事件流模型，并通过引入人机协同修复的方法解决了模型不够准确的问题。后续的工作会尝试结合图像处理等领域的技术，进行更精确的控件和路径识别，提高事件流模型的精确度。

参考文献

[1] R. Coppola, M. Morisio, M. Torchiano, Scripted GUI Testing of Android Apps: A Study on Diffusion, Evolution and Fragility, in: Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE, ACM, New York, NY, USA, 2017, pp. 22–32.

[2] A. M. Memon, M. L. Soffa, Regression Testing of GUIs, in: Proceedings of the 11th ACM SIGSOFT Symposium on Foundations of Software Engineering 2003 held jointly with 9th European Software Engineering Conferenc, Helsinki, Finland, September 1-5, 2003, pp. 118–127.

[3] N. Chang, L. Wang, Y. Pei, S. K. Mondal, X. Li, Change-Based Test Script Maintenance for Android Apps, in: Proceedings of 2018 IEEE International Conference on Software Quality, Reliability and Security, Lisbon, Portugal, July 16-20, 2018, pp. 215–225.

[4] M. Grechanik, Q. Xie, C. Fu, Maintaining and Evolving GUI-directed Test Scripts, in: Proceedings of the 31st International Conference on Software Engineering, May 16-24, 2009, Vancouver, Canada, Proceedings, 2009, pp. 408–418.

[5] A. Kervinen, M. Maunumaa, T. Pääkkönen, M. Katara, Model-Based Testing through a GUI, in: Proceedings of the 5th International Workshop, formal Approaches to Software Testing, Edinburgh, UK, July 11, 2005, Revised Selected Papers, 2005, pp. 16–31.

[6] S. Zhang, H. Lü, M. D. Ernst, Automatically Repairing Broken Workflows for Evolving GUI Applications, in: Proceedings of the international Symposium on Software Testing and Analysis, Lugano, Switzerland, July 15-20, 2013, pp. 45–55.

[7] W. Yang, Z. Chen, Z. Gao, Y. Zou, X. Xu, GUI Testing Assisted by Human Knowledge: Random vs. functional, Journal of Systems and Software 89 (2014) 76–86.

[8] A. M. Memon, Automatically Repairing Event Sequence-based GUI Test Suites for Regression Testing, ACM Trans. Softw. Eng. Methodol. 18 (2) (2008) 4:1–4:36.

[9] A. M. Memon, I. Banerjee, A. Nagarajan, GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing, in: Proceedings of the 10th Working Conference on Reverse Engineering, Victoria, Canada, November 13-16, 2003, pp. 260–269.

[10] Z. Gao, Z. Chen, Y. Zou, A. M. Memon, SITAR: GUI Test Script Repair, IEEE Trans. Software Eng. 42 (2) (2016) 170–186.

[11] X. Li, N. Chang, Y. Wang, H. Huang, Y. Pei, L. Wang, X. Li, ATOM: Automatic Maintenance of GUI Test Scripts for Evolving Mobile Applications, in: 2017 IEEE International Conference on Software Testing, Verification and Validation, Tokyo, Japan, March 13-17, 2017, pp. 161–171.

[12] F. Behrang, A. Orso, Test Migration for Efficient Large-scale Assessment of Mobile App Coding Assignments, in: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, Amsterdam, The Netherlands, July 16-21, 2018, pp. 164–175.

[13] M. B. Dwyer, V. Carr, L. Hines, Model Checking Graphical User Interfaces Using Abstractions, in: Proceedings of the 6th European Software Engineering Conference Held Jointly with the 5th ACM SIGSOFT Symposium on Foundations of Software Engineering, Zurich, Switzerland, September 22-25, 1997, Proceedings, 1997, pp. 244–261.

[14] G. Shah, P. Shah, R. Muchhala, Software Testing Automation using Appium, International Journal of Current Engineering and Technology 4 (5) (2014) 3528–3531.

[15] S. Singh, R. Gadgil, A. Chudgor, Automated Testing of Mobile Applications using Scripting Technique: A study on Appium, International Journal of Current Engineering and Technology 4 (5) (2014) 3627–3630.

[16] N. Verma, Mobile Test Automation with Appium, Packt Publishing Ltd, 2017.

[17] X. M. Wang, X. Huang, G. Z. Li, C. P. Miao, N. Shen, Research of Mobile Applications Automated Testing using Uiautomator, in: 2016 4th International Conference on Machinery, Materials and Computing Technology, Atlantis Press, 2016.

[18] N. Patil, D. Bhole, P. Shete, Enhanced UI Automator Viewer with improved Android accessibility evaluation features, in: International Conference on Automatic Control & Dynamic Optimization Techniques, 2016.

[19] 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.

[20] M. Macero, Macero, Anglin, Learn Microservices with Spring Boot, Springer, 2017.

[21] D. Zhang, Z. Wei, Y. Yang, Research on lightweight MVC framework based on SpringMVC and Mybatis, in: Proceedings of the 6th International Symposium on Computational Intelligence and Design, IEEE, 2013, pp. 350–353.

[22] K. S. P. Reddy, Java Persistence with MyBatis 3, Packt Publishing Ltd, 2013.

[23] L. Fu, Design and Implementation of Super Mall System Based on SOA Distributed Architecture, in: Proceedings of 2019 International Conference on Intelligent Transportation, Big Data & Smart Cit, IEEE, 2019, pp. 356–359.

[24] F. Junqueira, B. Reed, 谢超, ZooKeeper：分布式过程协同技术详解, 机械工业出版社, 2016.

[25] S. Li, X. Huang, Dubbo's Serialization Protocol Extension and Optimization of Its RPC Protocol Thrift, in: Proceedings of the 8th IEEE International Conference on Software Engineering and Service Science, 2017, pp. 721–726.

[26] Y. Fain, A. Moiseev, Angular 2 Development with TypeScript, Manning Publications Co., 2016.

[27] P. Japikse, K. Grossnicklaus, B. Dewey, Building Web Applications with Visual Studio 2017: Using. NET Core and Modern JavaScript Frameworks, Apress, 2017.

[28] L. Liu, Analysis and Application of MVVM Pattern, Microcomputer Applications 12 (2012) 019.

[29] F. Bao, J. Chen, Visual framework for Big Data in d3.js, in: IEEE Workshop on Electronics, Computer & Applications, 2014.

[30] Z. Yunliang, Z. Zhaofeng, Z. Xiaodan, X. Deshan, Web Dynamic Interactive Visualization of Knowledge Organization Systems with D3.js, Data Analysis and Knowledge Discovery 29 (7/8) (2013) 127.

[31] N. Q. Zhu, Data Visualization with D3.js Cookbook, Packt Publishing Ltd, 2013.

[32] L. Nair, S. Shetty, S. Shetty, Interactive Visual Analytics on Big Data: Tableau vs D3.js, Journal of e-Learning and Knowledge Society 12 (4).

[33] D. Wei, An Optimized Floyd Algorithm for the Shortest Path Problem, Journal of Networks 5 (12) (2010) 1496.

[34] Q. C. D. Do, G. Yang, M. Che, D. Hui, J. Ridgeway, Redroid: A Regression Test Selection Approach for Android Applications, in: Proceedings of the 28th International Conference on Software Engineering and Knowledge Engineering, Redwood City, San Francisco Bay, USA, July 1-3, 2016, pp. 486–491.

[35] B. Daniel, T. Gvero, D. Marinov, On Test Repair using Symbolic Execution, in: Proceedings of the 19th International Symposium on Software Testing and Analysis, Trento, Italy, July 12-16, 2010, pp. 207–218.

[36] A. M. Memon, An Event-flow model of GUI-based Applications for Testing, Softw. Test., Verif. Reliab. 17 (3) (2007) 137–157.

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

[38] B. Daniel, Q. Luo, M. Mirzaaghaei, D. Dig, D. Marinov, M. Pezzè, Automated GUI Refactoring and Test Script Repair, in: Proceedings of the 1st International Workshop on End-to-End Test Script Engineering, ACM, 2011, pp. 38–41.

[39] S. Huang, M. B. Cohen, A. M. Memon, Repairing GUI Test Suites Using a Genetic Algorithm, in: Proceedings of the 3rd International Conference on

Software Testing, Verification and Validation, Paris, France, April 7-9, 2010, pp. 245–254.

[40] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. D. Carmine, A. M. Memon, Using GUI Ripping for Automated Testing of Android Applications, in: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, Essen, Germany, September 3-7, 2012, pp. 258–261.

简历与科研成果

基本情况 陈笑智，男，汉族，1995 年 3 月出生，江苏省泰州市人。

教育背景

2017.9～2019.6 南京大学软件学院 硕士

2013.9～2017.6 中国药科大学理学院 本科

参与项目

1. 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。
2. 中央高校基本科研业务费专项资金资助：4-3基于可理解信息融合的人机协同移动应用测试研究（021714380017），2019.4-2019.12。

致　　谢

安卓GUI测试脚本修复系统从最初的想法到最后的实现历经了一年多的时间，在整个研究工作过程中我遇到了不少的问题，有思路上的也有实现上的，最终都得到了很好的解。在这里，我要对给予我关心和帮助的老师和同学表示诚挚的感谢。

首先感谢我的指导老师陈振宇教授，陈老师从项目的开题、研究思路、需求分析和系统设计等阶段给了我很多帮助和建设性的建议，对我的设计方案和实验思路给予很多有效的指正和指导。同时陈老师渊博的专业知识、勤恳的工作态度、勇往无前地探索精神也深深地感染着我，对我的研究生学习工作有非常高的激励和引导作用。

其次感谢实验室的房春荣老师、田元汉同学在本项目中付出的艰辛工作，项目的成功完成离不开大家的帮助和支持。感谢他们在我遇到问题时鼎力相助。

感谢我的家人在我的研究生学习期间对我的鼓励和支持，他们的关怀激励着不断向前，感谢每一位帮助过我、关心过我的老师、同学、朋友。

最后感谢于百忙之中抽出时间来参与我论文评审和答辩的专家们，向你们的辛勤劳动致敬!

版权及论文原创性说明

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

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

作者签名：

日期：　年　月　日