南京大學

研 究 生 毕 业 论 文

（申请工程硕士学位）

论　文　题　目　基于图像识别的跨平台测试脚本录制与回放系统的设计与实现

作　者　姓　名　赵文远

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：**MF1732180**

论文答辩日期：**2019 年 5 月 8 日**

指 导 教 师：　　（签字）

The Design and Implementation of Cross-platform Test Script Record and Replay System Based on Image Recognition

By

Wenyuan Zhao

Supervised by

Professor **Zhenyu Chen**

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

Software Institute

May 2019

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于图像识别的跨平台测试脚本录制与回放系统的设计与实现

工程硕士（软件工程领域） 专业 2017 级硕士生姓名： 赵文远

指导教师（姓名、职称）： 陈振宇 教授

摘 要

自动化测试具有高效率和高可靠等特点，逐渐成为移动应用开发过程中的主流测试方式。移动应用往往需要运行于多设备和多系统之上，设备和系统的“碎片化”问题使得现有自动化测试框架下测试脚本难以跨平台执行，从而导致测试脚本构建和维护成本很高。

本文设计与实现了一个基于图像识别的自动化测试脚本录制与回放系统，以解决测试脚本构建和维护成本高等问题。本系统模拟人工测试的流程，一方面提供设备远程操控，通过脚本录制回放方式简化测试脚本的开发过程，降低测试脚本开发技术门槛。另一方面脚本录制过程增加控件截图、控件布局位置等信息生成脚本，脚本回放过程通过图像识别技术对控件在新设备上进行定位，实现了脚本的跨设备和跨平台回放，避免了脚本的重复构建和维护。本系统主要分为设备管理模块、脚本录制模块与脚本回放模块。设计上对服务单独开发，分布式部署，模块间采用RESTful接口通讯，降低了模块间耦合，提高了系统的可扩展性。通过对ADB（Android Debug Bridge）与WDA （Web Driver Agent）工具封装，实现对Android与iOS设备的统一管理。充分利用MiniCap高速获取设备页面截图，通过Netty框架建立Web端与设备间TCP稳定传输通道，实现两端界面实时同步。充分利用SIFT、OCR（Optical Character Recognition）技术处理图片、布局信息完成控件定位，实现脚本跨平台回放。

本文选取了10台主流移动设备与5款移动应用对系统可用性进行分析验证，实验结果表明本系统录制流程对Android平台下不同设备有较好的支持度，脚本跨设备回放成功率80%以上，跨系统回放成功率60%左右。本系统已部署用于公司内部移动应用测试，通过测试脚本单次录制跨平台回放，有效减少了测试脚本维护数量，降低了测试成本。

关键词： 移动自动化测试，GUI自动化测试，录制回放，跨平台，SIFT，OCR

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 Cross-platform Test Script Record and Replay System Based on Image Recognition

SPECIALIZATION: Software Engineering

POSTGRADUATE: Wenyuan Zhao

MENTOR: Professor Zhenyu Chen

Abstract

Automated testing is characterized by high efficiency and high reliability, and has gradually become the mainstream testing method in the development of mobile applications. Mobile applications often run on multiple models and multiple systems. The "fragmentation" of devices and systems makes it difficult to implement cross-platform execution of test scripts in existing automated testing frameworks, resulting in high cost of test scripts build and maintenance.

This thesis designs and implements an automated testing script record and replay system based on image recognition to solve the problems of high cost of test scripts build and maintenance. The system simulates the process of manual testing. On the one hand, it provides remote control of the device, and the development process of test scripts is simplified by the script record and replay mode, which reduces the technical threshold of test scripts development. On the other hand, the script record process generates scripts by adding information such as screenshot and layouts position of the component. The script replay process uses the image recognition technology to locate the component on the new device, which realizes cross-device and cross-platform replay of the script, thereby avoiding scripts of repeating builds and maintenance. The system is mainly divided into a device management module, a script record module and a script replay module. In design, the services are developed separately, distributed deployment, and RESTful interfaces are used for communication between modules, which reduces the coupling between modules and improves the scalability of the system. Through the encapsulation of ADB (Android Debug Bridge) and WDA (Web Driver Agent) tools, unified management of Android and iOS devices is realized. The system makes full use of the MiniCap tool to obtain screenshots of the device page at

high speed, and establishes a TCP stable transmission channel between the web and the device through the Netty framework to realize real-time synchronization of the interfaces at both ends. The system makes full use of SIFT and OCR (Optical Character Recognition) technology to process pictures and layout information to complete component positioning and achieve cross-platform replay of scripts.

This thesis selects 10 mainstream mobile devices and 5 mobile applications to analyze and verify the system availability. The experiment proves that the record process of this system has better support for different devices under the Android platform, and the script cross-device replay success rate is over 80%. The cross-platform replay mode has a success rate of about 60%. The system has been deployed for internal mobile application testing. Through a single record of test scripts and cross-platform replay of scripts, it effectively reduces the number of test script maintenance and the cost of automated testing.

Keywords: Mobile Automated Testing, GUI Automated Testing, Record and Replay, Cross-Platform, SIFT, OCR

目　录

表　目　录

图　目　录

第一章　引言

1.1　背景与研究意义

随着互联网的不断普及，特别是智能手机诞生、大力普及以来，移动互联网在互联网中所占的比重持续不断的攀升。据2018中国互联网信息中心统计数据显示，2018上半年移动互联网网民增长人数高达3509万，手机网民数量占总体网民数量比重达到98.3%。各大手机厂商为满足不同用户的需求，推出了各种不同的设备机型，仅2018年1月至11月上市的智能手机即达到551款。移动互联网的飞速发展，为移动应用软件的发展带来了巨大的机遇。截止2018年7月底，仅中国市场上监测到的移动应用数量即为424万 [1]，许多类别的移动应用下载数量已超过千亿次，移动应用数量正以井喷式增长。移动互联网已经成为了互联网领域最具有活力的分支。

随着移动互联网的全面、快速发展，众多的移动互联网用户对移动应用软件的需求不再仅仅局限于功能层面，人们对应用软件的质量变得愈加重视。作为移动应用软件从开发到交付用户这一过程中的质量保证，软件测试在整个软件的开发过程中都显得尤为重要。目前大多数企业对软件的测试分为单元测试、接口测试、UI（User Interface）测试三个层次。由于单元测试、接口测试等相对较为稳定，现有的测试框架已较好的满足了两者测试的需求。移动应用程序版本的频繁迭代使得大量的测试资源消耗在回归测试中，而UI测试作为回归测试中重要组成，对UI测试效率进行优化提升，较大程度上影响了移动应用测试成本 [2]。

目前业界中UI测试通过人工测试和编写自动化测试脚本两种方式实现 [3]。移动应用自身具有多平台、多设备运行的特点，这些都使得应用软件在UI测试方面存在着大量的重复性工作。人工测试方面，由于测试人员的主观性比较大，测试的准确性较大的依赖于测试人员自身，大量的测试用例往往带来的结果是测试不完整、缺陷记录不详细、缺陷难以复现等问题。而现有的软件开发较多的采用了敏捷式迭代，效率低下的人工测试方式极大影响了软件开发的进程。测试开发人员编写自动化脚本方式，在一定程度上提高了移动应用的测试效率。但由于移动设备机型及平台多样性，UI界面自身的多变性，使得UI自动化测试脚本在实际生产过程中存在大量重复性修改、维护 [4]。同时测试脚本的编写基于高级语言来进行，对于测试人员有着较高的技术要求。不同平台、机型下的大量重复性脚本需要修改和维护，也导致了企业测试的成本较为高昂。

对比人工测试与现有自动化测试的不同可以发现，人工测试的角度是基于对UI界面的识别、操作而完成测试。从该角度出发，将编码式的脚本开发简化为操作式的脚本录制，即基于Web实现远程设备操控，点击应用控件，后端自动提取控件图片、布局位置、XML文件等属性生成测试脚本。脚本回放阶段借助图像识别技术，结合脚本信息实现对控件在不同设备界面上的定位，获取坐标信息后借助底层工具实现对脚本操作的回放。这样从模拟人工测试角度出发，能够更好的解决测试脚本的多机型、多平台重复编写问题，实现一次脚本录制，多机型、多平台回放测试，极大减少了脚本开发、维护的复杂度。相比传统自动化测试中不同平台下采用不同语言编写测试脚本代码，脚本点击录制方式也更易于测试人员接受 [5]。

综上所述，在移动自动化UI测试过程中，由于移动设备种类多样性及系统平台的差异性，人工测试与基于编码的自动化测试方式均存在着成本高、难维护等缺点。因此设计一种能够简化脚本开发、提高测试脚本跨平台能力的自动化测试系统或框架是有意义且十分必要。通过本系统，测试人员能够便捷的录制脚本，同时脚本能够支持跨机型、跨平台回放，提高了自动化测试脚本的跨平台能力，进而提高测试效率，缩短测试周期，降低了企业在移动应用测试上的成本。如何借助图像识别等技术实现上述系统即是本文研究与阐述的重点。

1.2　国内外研究现状

软件自动化测试从起步时借助硬件来记录用户操作流程进行回放来实现自动测试；到测试脚本阶段，专业的测试开发人员通过复杂的脚本语言编写、维护测试脚本来实现自动化测试；到目前的测试框架阶段，出现大批的测试框架，在一定程度上优化了脚本的编写，加速了脚本的生成，提高了可维护性，使得软件测试效率得到一定程度的提升。

同时，根据Gartner研究机构数据表明，在全球出售的智能手机中，99.9%的移动设备使用的为Android或iOS系统，Android系统的全球份额达到了85.9% [1]。目前主流的移动自动化测试框架基本是针对Android、iOS两款系统来设计的。Android平台方面主流自动化测试框架包括Instrumentation[1] 、Robotium[2] 、UIAutomator[3] 、Espresso[4] 等，iOS平台主流自动化测试框架为KIF[5] 、UIAutomation[6]

[1]https://developer.android.com/reference/android/app/Instrumentation

[2]https://github.com/RobotiumTech/robotium

[3]https://developer.android.com/training/testing/ui-automator

[4]https://developer.android.com/training/testing/espresso

[5]https://github.com/kif-framework/KIF

[6]https://developer.apple.com/library/ios/documentation/DeveloperTools/Reference/UIAutomationRef/

等 [6]，Appium[7] 则是支持两者的跨平台测试框架。常用移动自动化测试框架特征对比如表 1.1 所示。

表 1.1: 常用移动自动化测试框架对比

特征 \ 框架	Robotium	UIAutomator	Espresso	UIAutomation	Appium
支持平台	Android	Android	Android	iOS	Android、iOS
控件捕获成本	较高	较高	较高	较高	高
跨平台	否	否	否	否	是
跨应用	否	是	否	是	是
支持Hybrid	支持	不支持	支持	支持	支持
稳定性	中	中	中	中	中

Instrumentation框架是Android自带的一个测试框架，因此成为了很多自动化测试框架实现的基础，其具有很多丰富的高层次封装，避免了使用者过多的二次开发。但是Instrumentation不支持跨应用，因此基于其实现框架都继承了这个缺点 [7]。Robotium便是基于Instrumentation 封装后得到的一个更强的框架。对常用的操作在易用性方面进行了封装、改进，多用于功能性、系统和验收测试场景。但是不能够支持跨应用、跨平台或系统测试。UIAutomator是Google支持、维护的测试框架，能够支持Android 4.1及以上版本系统，该框架更好的支持Java语言开发，能够支持跨应用测试，但并不支持跨平台和Hybrid应用。Espresso 同样是Google开源的测试框架，其具有规模小、更简洁、易编码、易上手等特点，基于Instrumentation开发，不能够很好的支持跨应用、跨平台测试。

iOS平台由于自身生态的封闭性，且其在全球份额占比相比较少，自动化测试框架方面相比Android平台数量较少。KIF(Keep It Functional) 是一款iOS应用功能性测试框架，使用Objective-C编写，对于iOS开发者较为易于上手，但其缺点运行较慢，不支持跨平台测试。UIAutomation 是Apple提供的UI自动化测试框架，使用Javascript编写，能够较为精准的查找和操控控件元素，能够支持脚本的录制与回放，但UIAutomation在自动化测试过程中出现崩溃便终止测试，降低了测试的整体效率，不支持跨平台。

Appium 测试框架是能够支持跨平台的传统的移动自动化测试框架，能够支持原生应用、Web应用、Hybrid应用的测试，且能够同时支持Android、iOS平台及Firefox浏览器，无需对被测应用进行编译或调整，非侵入式的完成自动化

[7]http://appium.io

测试 [8]。但Appium自身便是对不同框架的封装，且由于其基于WebDriver协议的方式，使得测试代码的维护性较之其他框架更差 [9]。

通过对以上测试框架比较可以得出，传统的自动化测试框架主要存在跨平台能力、跨应用能力差 [10]、控件捕获成本较高、脚本的稳定性较大程度上依赖于控件的Id是否改变，以及测试脚本的开发、维护需要一定的编程基础，门槛相对较高等不足。在此背景下，模拟人工测试流程的所见即所得的测试框架Sikuli、AirTest通过图像处理技术对控件截图在界面中进行匹配定位，实现脚本回放，带给我们对于自动化测试不同的思考 [11]。

Sikuli [8]是早期PC时代的产物，是MIT研究团队发布的以图像检索技术为基础的图形化编程技术 [12]。测试人员可以利用GUI 对象的屏幕截图作为函数的参数直接引用进行脚本编写，简化了以往测试脚本编码方式，采用操作与截图组合的脚本语言风格，使得脚本编写变得更为简单、直观。脚本回放阶段，利用图像检索算法对控件截图在当前屏幕中匹配相对应的控件，并对其应用相应的操作 [13]。Sikuli基于图像处理的思路使得其具有支持跨平台、跨应用、支持Hybrid等特性 [14]。但由于当时图像处理技术的准确率较大程度上限制了框架的可用性。对于图片的相似度调整仍需要手动的单个进行，工作量大等等原因限制了Sikuli 实际的应用与推广。

对于国内来说，网易推出的Airtest [9]是一款基于图像识别和Poco控件识别的一款跨平台UI自动化测试框架。Airtest是网易团队自己开发的一个图像识别框架，整个框架就是基于Sikuli的图像识别思想进行改进。同时基于Poco控件搜索框架，Poco框架是网易自主研发的UI测试框架，原理类似于Appium，通过控件的XPath、Id等来定位目标控件，然后调用函数方法对目标控件进行点击或操作。Airtest是目前较为完善的跨平台自动化测试工具，但该工具主要应用对象是游戏等应用，对于跨平台部分实践中仍存在较多问题。

1.3　本文主要研究工作

通过对目前常用的移动应用GUI自动化测试框架研究可以发现存在两点主要问题：多数框架目前仍借助各种高级语言来编写自动化测试脚本，提高了自动化测试人员使用框架的技术门槛；测试脚本跨应用、跨设备和跨平台测试能力不足。

本文设计实现了基于图像识别的跨平台脚本录制与回放系统。针对传统移动自动化测试框架中存在的编码技术门槛高，跨应用、跨平台能力差，脚本维

[8]http://www.sikulix.com

[9] http://airtest.netease.com

护困难等问题提出相应解决方案。一方面采用录制-回放的方式简化了脚本生成过程，降低了脚本生成的技术门槛。另一方面结合图像处理技术实现脚本的单次录制，多设备、多平台运行回放，强化了系统的跨平台能力，进而避免了传统自动化测试框架下同一应用因不同机型、平台造成的脚本的重复编写，减少了脚本的维护数量。系统基于B/S模式，测试人员可以通过浏览器访问远程真机界面，按照测试用例流程操作应用界面即完成脚本的录制，之后选择录制的脚本与回放设备实现对应用在不同平台和不同设备下的回归测试。

本文主要通过录制与回放形式简化了脚本开发过程，通过图像处理方法提高了测试脚本跨应用、跨平台执行能力，进而提高了脚本的复用性。为此本文在设计实现系统时从以下两个方向进行改进与优化。

（1）便捷、精准的脚本录制方式。对传统框架下控件的定位方式进行补充，增加控件截图、布局位置信息构建图形化增强控件模型。向测试人员提供Web端交互界面，通过MiniCap等实现真机页面远程实时同步交互。用户仅需按照测试流程流程点击界面控件元素，后端自动完成操作记录，脚本实时展示到Web端，完成录制后对脚本进行保存。整个过程与用户使用终端人工测试相仿，降低了脚本测试的门槛，更易于测试人员接受，上手简单，一定程度上提高了脚本录制的效率。

（2）多方案实现脚本跨平台回放。回放过程即借助图像信息、传统控件属性对控件在新设备上进行定位，输出新设备之上的物理坐标，进而生成指令操作手机。为了保证录制脚本的稳定性、可用性、跨平台特性，采用图像匹配与布局匹配两种方法实现控件的定位，完成脚本回放。

图像匹配方法通过脚本中控件截图与新设备当前页面截图之间进行图片特征匹配，完成对控件的位置定位。布局匹配方法作为图像匹配的完善补充，借助图像分割、OCR技术等对页面截图进行布局刻画分析，输出操作控件在页面的布局位置坐标。对回放设备页面截图同样采用布局刻画，通过布局位置坐标最终实现控件的定位。

1.4　本文的组织结构

本文共分为五个章节，组织结构如下：

第一章 引言。本章主要介绍了跨平台的脚本录制回放系统的研究背景及意义，国内外移动应用自动化测试框架的发展现状，并对本文主要研究工作进行了阐述。

第二章 相关概念及技术。本章首先对自动化测试及移动应用类别相关概念

进行了阐述，之后对项目所涉及到的技术、框架进行了介绍，其中包括设备管理工具WDA（Web Driver Agent）与ADB （Android Debug Bridge）、SIFT特征匹配算法、Tesseract-OCR技术、WebSocket协议、Netty框架等。

第三章 跨平台测试脚本录制与回放系统的需求分析与概要设计。本章首先对项目整体背景进行描述，分析和总结出系统的功能性与非功能性需求，并结合用例图和用例表对需求进行阐述。给出了系统总体架构与设计思路，对系统涉及的设备管理模块、脚本录制模块及脚本回放模块的设计方案进行说明。最后对系统数据库设计结合ER 图进行了阐述。

第四章 跨平台测试脚本录制回放系统的详细设计与实现。在第三章中对系统的需求分析及概要设计的基础上，通过展示调用顺序图、类图及关键代码分别对对各个模块主要功能实现细节进行阐述。设备管理模块主要包括设备接入与状态更新与设备数据传输，脚本录制模块主要包括脚本单步录制操作记录与脚本生成，脚本回放模块主要包括脚本单步操作回放、图像匹配定位与布局匹配定位。最后将系统部署至测试环境，对系统进行了单元测试及功能测试，并设计实验对系统运行效果进行分析，给出系统实际运行界面的展示。

第五章 总结与展望。本章总结了论文完成期间所做的工作，并就跨平台的脚本录制与回放系统的未来扩展提出了进一步的展望。

第二章　相关概念及技术

2.1　自动化软件测试

软件的自动化测试是将以人为主体进行驱动的测试行为转化为机器按照测试脚本流程自动执行测试的一种过程，即通过工具执行由程序语言编制的测试脚本模拟手工测试的步骤来实现对软件的自动测试 [15]。自动化测试之所以在软件开发的过程中逐渐取代了人工测试，这是由软件开发自身开发周期越来越来短，且较之人工测试，自动化测试带来了更多成本、时间及质量等方面上的优点 [16]。具体优势如下：

便于回归测试：自动化测试最主要的用处便是回归测试，尤其在软件开发迭代迅速，程序修改较为频繁的时候自动化测试在回归测试的优势更为明显。回归测试主要是通过以往测试用例验证软件变动是否对现有功能产生影响，因此回归测试的自动化可以明显的提升测试效率，缩短测试时间。

一致性和复用性：由于自动化测试大多是通过编写测试脚本，由框架运行脚本来实现，因此测试脚本每次执行内容的一致性可以得到保障。在软件功能不变动或变动不大时，仅需对现有测试脚本进行少量修改即可实现测试用例的覆盖。测试脚本的开发可以在原有测试脚本的基础上修改完成开发，具有一定的复用性 [17]。

合理的配置资源：自动化测试来完成以往人工测试的工作，将更多的测试人员从枯燥、重复的测试工作中解放，可以使得更多的测试人员投入到脚本开发、维护方面，优化了测试资源。

执行复杂测试：自动化测试按照程序流程执行，因此可以在更短的时间内更精准的完成复杂的、人工难以完成的测试。如软件进行性能测试时需要大量的用户同时使用，而一般企业中难以实现，因此可以借助自动化测试工具模拟用户操作，完成对软件的测试。

增强软件可信度：由于自动化测试是机器严格按照测试脚本执行的，因此不存在人工测试中的疏忽或错误情况，完全取决于测试用例的设计好坏。经过完善自动化测试后的软件在可靠性上也会大幅提升。

软件自动化测试极大程度上提高了软件测试效率，缩短了软件开发的迭代周期，为软件质量提供了强有力的保障。移动应用自动化测试是软件自动化测试在移动应用领域的应用。由于移动应用需要运行于多机型，不同平台之上，

测试开发人员需要在现有自动化测试框架下针对不同平台分别进行测试脚本的开发与维护，平台的差异性使得不同平台下测试脚本不能跨平台执行 [18]，因此测试人员需要掌握不同平台脚本开发语言，为单个应用开发、维护多组测试脚本。移动应用自动化测试中测试脚本开发对测试人员技术要求高，测试脚本难以实现跨平台执行正是本系统需要解决的问题存在。

2.2　移动应用类别

移动应用是指运行在移动终端设备之上的应用软件，由于移动系统平台、终端设备、用户场景等种类繁多，为了更好的满足各种场景下的需求，移动应用逐渐发展为多种类型。目前主流的移动应用程序大体分为三类：原生应用(Native App)、网页应用(Web App)、混合应用(Hybrid App) [19]，具体关系如图 2.1所示。移动应用种类、运行平台的多样性为移动应用的自动化测试带来了跨平台的复杂性问题。在很多大型互联网公司，为了在不同的场景下获取用户，往往会选择以多种方式开发同一款应用，为培养用户使用习惯，这些不同平台下应用在功能、界面上都基本相同 [20]。因此更好的了解移动应用的类别，对不同类别及平台下应用业务流程及特点进行对比分析，在解决移动应用跨平台及脚本重复开发等问题上有着重要意义。

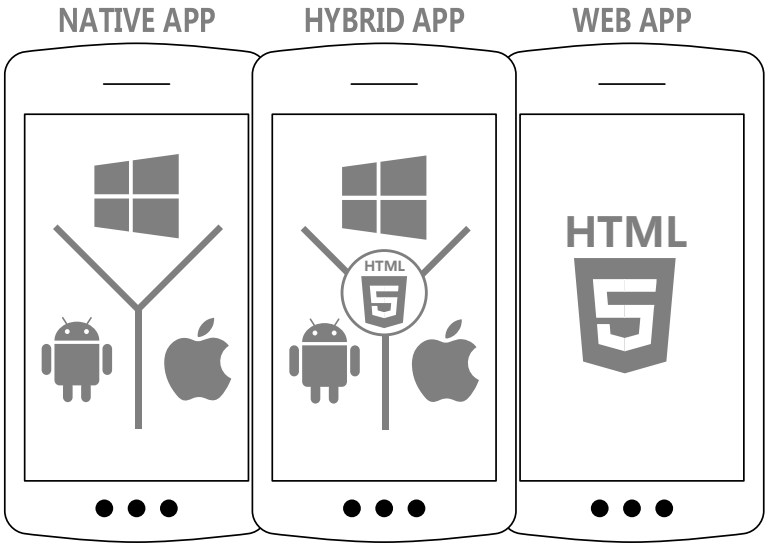

图 2.1: 移动应用类别关系图

原生应用(Native App)：是指基于移动终端自身操作系统使用系统原生提供的开发工具等编写的第三方移动应用程序，也被称之为本地App。在不同

的系统平台下基于不同语言开发，如Android平台基于Java语言，iOS平台基于Object-C语言。直接运行于系统平台层上方，系统兼容性强，应用可定制化程度高，相比其他模式能够提供更好的用户体验，响应速度快。但由于版本升级困难，且应用需要用户安装后才能使用，造成了应用整体的开发、维护成本的较高，且应用不能跨平台运行。不同的平台会提供不同的自动化测试方案，测试开发人员往往需要针对不同系统平台使用不同语言开发相应的自动化测试脚本完成对原生应用的自动化测试 [21]。不同系统平台测试脚本不能进行跨平台执行。

网页应用(Web App)：指采用网页技术开发出的移动应用，依托于终端浏览器为用户提供服务，无需下载安装即可使用。相比之下，该类应用开发成本低，版本更新便捷，支持跨平台、跨设备运行 [22]。但该种应用强依赖于浏览器，用户体验不如其他模式，且功能受限严重，大量移动端功能无法实现。目前较为流行的微信“小程序”即为该类应用的代表。网页应用能够在不同系统平台的浏览器中运行，与系统平台关联性较弱。对该类型应用测试的本质是对不同浏览器进行测试，多数Web测试技术可以应用到此类应用测试。

混合应用(Hybrid App)：是指原生与Web两种模式混合开发的移动应用。JS调用原生统一提供的API接口，结合H5实现应用主要逻辑的开发，最终在UIWebView中显示。使用该种方式，开发者可以通过H5、JS语言即可开发在不同平台下可部署运行的类原生应用 [23]。在兼顾性能与用户体验的同时节省了应用开发的时间和成本。由于混合应用是原生与Web两种类型的混合，因此对于混合应用的测试较为复杂，应用原生部分测试需要根据不同系统编写相应测试脚本完成测试，Web部分则可以通过现有Web测试技术实现跨系统测试，使得混合应用同样面临着跨平台测试的问题。

2.3　移动设备管理工具

2.3.1　WDA

WDA全称Web Driver Agent，是由Facebook推出的iOS平台下的一款开源移动测试框架。WDA采用WebDriver协议，专为iOS平台设计实现，可用于iOS 设备的远程控制 [24]。它支持应用程序的远程启动与终止，同时能够覆盖点击、滚动及获取页面元素等多数设备操控形式，是目前iOS平台通用设备自动化及应用端到端测试方面表现较好的测试框架。系统中使用WDA框架建立设备管理服务与iOS平台设备的连接，通过对WDA设备操控服务进行接口封装，为iOS设备的管理与操控问题给出合适的解决方案。

WDA初始化阶段会在设备端安装WebDriverAgent服务，通过HTTP请求实现与服务端通讯，完成客户端命令发送与结果获取。WebDriverAgent服务与XCTest.framework[1]建立连接并调用底层API实现命令在设备上的执行。通过该交互方式，WDA仅需在设备上初始化Server服务即可无侵入的进行设备的管理及跨应用UI测试 [25]。WDA具有以下优势：

支持真机：支持iOS平台模拟器与真机设备。方便通过模拟器进行开发调试，同时为实现真机iOS设备管理提供了保证。

支持USB通信：USB通信解决了无线网络设备与服务通讯不稳定问题。iOS设备通过USB连接到Mac服务器在保证通信效率的同时提高了通信的稳定性与可靠性。测试过程中服务器通过USB能够为设备进行充电，解决了设备的测试续航问题。

可用性：通过WebDriver规范对iOS设备常用操控形式进行封装，支持iOS系统下的多机型及多版本系统运行，支持多数场景下应用的测试。

2.3.2 ADB

ADB全称Android Debug Bridge，即安卓调试桥。ADB是Android SDK（Software Development Kit）自带一款调试工具，是Android设备与PC端连接的桥梁 [26]。ADB采用标准C/S架构，包含运行于服务器上的Client端与Server端，以及运行于设备上的Deamon守护程序。服务器Client端负责发送操控命令与运行结果展示，Server端实现对Client端与Deamon程序的管理与消息中转，Deamon守护程序接收到操控命令进行解析并对相应设备完成操控 [27]。ADB能够无侵入完成设备管理与应用测试。系统通过ADB工具实现了设备管理服务对Android设备的集中化管理，包括对设备信息、状态查看及设备操控，为脚本录制与回放提供了基础支持与保障。ADB主要具有以下特点：

支持Shell命令：提供Shell命令获取设备信息或进行设备操控，方便服务器端进行接口封装，为上层服务提供设备管理操控接口。

多版本系统支持：能够实现对Android多版本系统的支持，在设备管理服务设计开发时避免了系统版本多样性为服务开发引入的复杂性及问题。

支持USB通信：支持服务端与Android设备通过USB方式连接通信，采用TCP协议避免了网络带来的延迟与不稳定性问题 [28]。USB连接能够在保证设备进行测试的同时为设备提供电力保证，解决了设备续航的问题。

[1]https://developer.apple.com/documentation/xctest

2.4　SIFT特征匹配算法

SIFT是D.Lowe提出的一种局部特征描述算子 [29]，主要用于处理图像间存在的平移、旋转及仿射变换情况下的匹配问题。SIFT描述算子能够保持很好的不变性因此具备很强的匹配能力。SIFT特征匹配算法则是基于该描述算子的一种提取局部特征的模式识别算法技术。

SIFT算法包括两个处理阶段，第一阶段是SIFT特征点的生成，即根据SIFT描述算子的定义从图像中提取对尺度缩放、旋转、亮度等变化无关的特征向量；第二阶段是对提取到的特征点进行匹配筛选，即从两组特征点中找出距离低于某个阈值的特征点，认为特征点匹配成功。SIFT特征匹配算法已在长期的实践被证明为同类算法中最具健壮性的算法，已被广泛应用到各种图像匹配的场景之中，是目前国内外研究的热点之一。SIFT算法主要具备以下优势：

局部特征不变性：SIFT算法提取的是图像的局部特征向量，且SIFT算子对图像旋转、亮度变化、尺度放缩均具有不变性的特点，此外对视角变化、仿射变换、噪声等也具有很好的稳定性 [30]。

多量性：SIFT算法即使在图像仅有少数物体时也能够提取出大量的SIFT特征向量用于图像的匹配。

高速性：SIFT算法计算速度较快，能够满足大多数非实时性场景的需求。经过改良后的SIFT算法甚至可以达到实时的要求 [31]。

可扩展性及易用性：SIFT算法提取出的特征向量可以很便捷的与其他形式的特征向量联合使用，具有较强的扩展性。且该算法处理过程清晰，不需要过多的使用经验即可快速上手，易于开发。

本系统脚本回放阶段通过测试脚本中控件截图与回放设备界面截图进行匹配、定位控件，并反馈匹配区域坐标位置。该应用场景下利用SIFT特征匹配算法能够高准确率、实时的进行控件定位，满足系统对匹配响应时间的要求。算法具有的旋转、平移等变换不变性则进一步提高了匹配准确率，降低了开发的复杂度。

2.5　Tsseract-OCR技术

OCR(Optical Character Recognition)，又被称之为光学字符识别，是指电子设备对图像文件上的字符进行分析处理，通过字符识别方法获取文字和版面布局信息的过程 [32]。Tesseract是由最早由惠普实验室提出并研发的一款专利软件，到1995年已经成为OCR业界最精准的识别引擎之一。随后由于惠普放弃该

项目，直至2006年将其开源后由Google接手对其进行改进、优化、消除Bug后免费提供外界使用。

Tesseract是目前开源OCR引擎中识别精度最高的引擎之一 [33]。截止目前Tesseract 4已可以“开箱即用”式支持100多种语言的识别，并能够支持纯文本、HTML、PDF等多种格式的输出。为了提供更高的识别精度，Tesseract提供了一套可供用户自主训练字符库的训练方法。具有可多平台运行、开源、轻量、支持自主训练字符库、支持版面分析等的优点。

Tesseract能够支持图片布局分析和字符分割与识别。本系统脚本回放布局匹配过程采用Tesseract对录制界面及回放界面预处理结果分别进行布局切割，通过布局比较定位控件。Tesseract布局分析进一步提高了图片Canny布局刻画结果的准确度，从而提高了布局匹配的可靠性。

2.6　Netty框架

Netty是一个基于异步事件流驱动的网络通信应用程序框架，可以用于对可维护的高性能等的协议服务器与客户端的迅速开发 [34]。Netty吸取了以往多种协议实现时的经验如FTP、SMTP、HTTP以及多种二进制和基于文本传输的协议，在此基础上对Netty 进行了精心的设计。因此Netty是一种同时可以兼顾性能、稳定性、灵活性，同时又极大的降低了底层通信开发复杂度的框架程序 [35]。Netty框架已在互联网、大数据、网络游戏、电信软件等众多行业经历了大规模的商业应用考验，框架的质量得到了充分的验证。Netty 框架分别在设计、易用性、性能方面具有以下特点：

设计：采用了统一的API，可以面向各种不同类型传输形式进行开发，如阻塞、非阻塞。采用了灵活、可扩展的事件模型使用框架进行系统的开发，允许关注点的分离。具备高度可定制化的线程模型，支持单线程、一或多个线程池的使用。数据包套接字给出了真正的没有连接的支持。

易用性：框架API使用简单，具有较低开发门槛，同时给出了详尽的Javadoc文档、用户使用指导及相应的代码示例。框架不需要额外的依赖关系，Netty 3.X仅需JDK 5、Netty 4.X仅需JDK 6的支持。

高性能：Netty框架在提高了系统吞吐量的同时降低系统响应的延迟。降低了对客户端、服务器等的资源消耗，使不必要的内存副本降至最小。

通过使用Netty框架，可以极大的简化NIO网络编程，开发人员无需关心线程的管理与线程间的通讯实现，更多的将精力投放到业务逻辑与实现。在保证开发效率的同时能够使系统的性能达到更高，稳定性变得更强。由于Netty基于

事件驱动的设计思想，使得每个单一的业务逻辑功能都有相应的函数实现，对于程序的可扩展性与维护性具有极大的提升。

2.7　WebSocket协议

WebSocket是HTML5所定义的一种新的通讯协议。主要用于解决Web浏览器和服务器之间进行任意的双向数据传输时HTTP协议不支持的问题。WebSocket协议基于TCP协议实现，主要包括初始时借助HTTP协议完成握手过程，及建立TCP连接后的多次数据帧双向传输过程。WebSocket成功解决了Web客户端与服务器进行频繁双工通讯的情况，有效的避免了反复建立HTTP连接时带来的资源浪费问题，提高了通讯效率及资源利用率 [36]。传统HTTP 与WebSocket通讯交互比较如图 2.2所示：

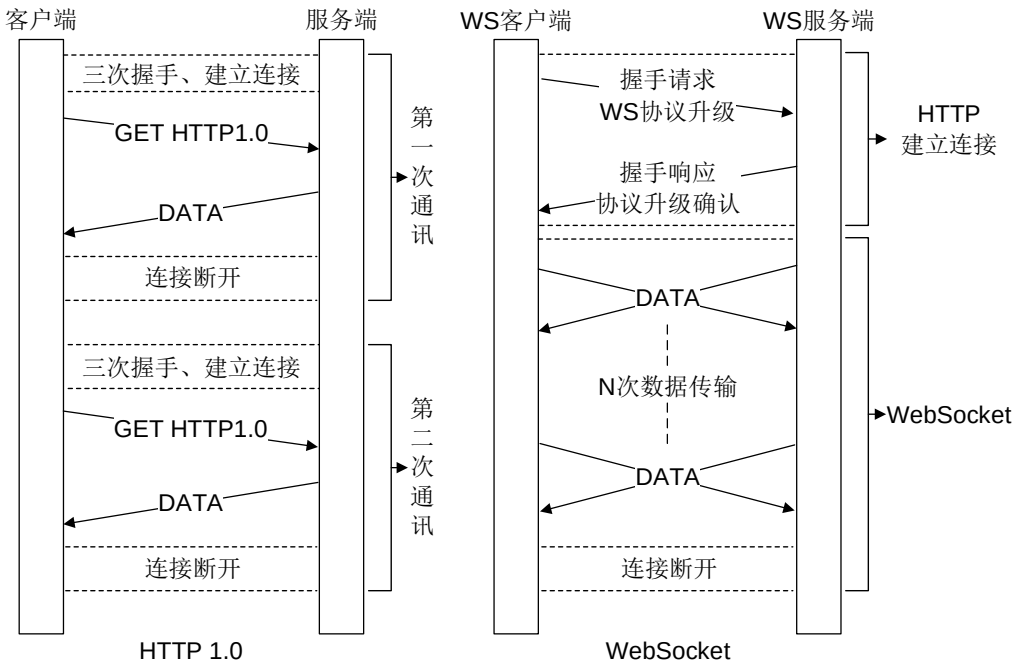

图 2.2: HTTP与WebSocket通讯对比图

WebSocket具有实时性高、节约带宽等优点 [37]。

实时性高：传统轮询方式下如果设置间隔时间较短会对服务器造成较大的压力，因而轮询方式时间间隔周期一般较长，实时性差。WebSocket是由服务器主动进行信息推送，实时性较高。

节约带宽：轮询方式是基于HTTP协议，header信息量较大，有效数据占比较低，造成了带宽的浪费，而WebSocket方式，header信息量较小，有效数据占比高，进而节约了带宽。

本系统中通过WebSocket实现浏览器与后端服务之间的TCP双工通信，使设备界面图片能够通过TCP进行稳定、低延时的传输，保证了前端与设备界面的实时同步，提高系统设备远程操控的可用性。

2.8　本章小结

本章主要对系统中涉及的相关概念及使用到的工具、框架等进行介绍，并结合工具、框架优点阐明选用理由。首先介绍了软件自动化测试相关概念，并与人工测试方式进行对比，阐述了自动化测试的优势，指出自动化测试在移动应用方面应用仍存在的不足。随后对移动应用的类别进行简要介绍，指出应用类型为脚本跨平台、跨设备方面带来的阻碍。最后依次对系统中涉及的技术、框架等做了介绍。WDA、ADB工具，主要用于iOS、Android设备的管理操控。SIFT特征匹配算法，主要用于图像匹配实现控件定位过程。Tesseact-OCR技术，主要用于布局匹配实现控件定位过程。WebSocket协议与Netty框架结合构建设备到前端的TCP数据传输通道。

第三章　跨平台测试脚本录制与回放系统需求分析与概要设计

3.1　项目整体概述

伴随智能手机终端设备在社会上的广泛推广，移动应用已经成为大家生活中不可或缺的重要组成部分，在人们的社交、学习、娱乐生活中发挥着巨大的作用，为我们的生活提供了极大的便捷。作为移动应用好坏与否的重要考量因素——质量，渐渐成为用户在众多同类软件中选择使用哪款的重要条件。由于移动设备平台、类型繁多，如何保证多种设备下应用的质量始终是移动应用软件的一大重难点问题。

如今各大公司为使移动应用在最终投放，交付用户使用前质量能够得到保障，纷纷采用测试开发人员使用编程语言通过自动化测试框架编写、维护测试脚本对软件进行测试的方式。然而在目前众多自动化测试框架下，脚本的编写需要根据应用版本、运行平台、手机类型等开发，往往导致同一功能需要编写、维护多份自动化测试脚本，带来测试周期过长，测试成本高昂等现实问题。

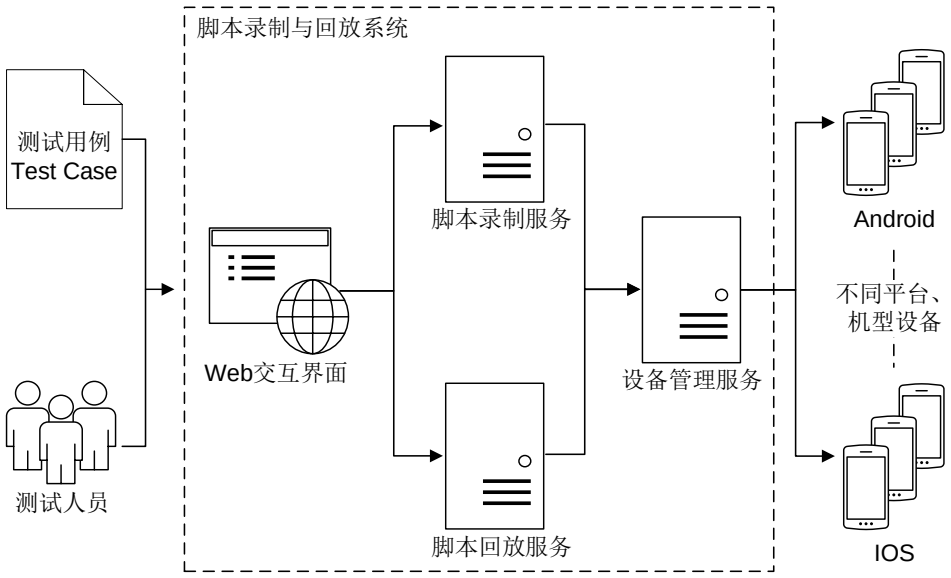

图 3.1: 脚本录制回放系统的应用场景图

本文设计实现的测试脚本录制与回放系统，将测试人员、测试用例与待测应用相关联，系统的具体应用场景如图 3.1所示。本系统主要面向不具备或具有

少量编程技能的测试人员使用，系统提供Web端交互界面，将远程真实设备界面投放到Web端供测试人员操控。系统包括脚本录制服务、脚本回放服务、设备管理服务，测试人员使用测试录制服务，将测试用例通过Web端设备界面操作完成相应脚本的录制。将录制的测试脚本使用脚本回放服务选择不同平台、不同类型的设备完成脚本的回放。设备管理服务主要接收来自录制、回放服务的终端命令传递至具体终端，对设备状态进行管理，同时将设备当前界面向上传输。本系统脚本的跨平台、跨设备回放是借助图像识别技术实现。

3.2　录制回放系统需求分析

3.2.1　功能性需求

跨平台脚本录制与回放系统的主要功能是供测试人员便捷的进行测试脚本的录制及实现录制的脚本在不同设备上的回放运行。对系统的功能需求按照设备管理、脚本录制与脚本回放三个模块进行分析。设备管理模块通过USB与移动设备直接连接，主要负责移动设备接入、状态管理、设备操控及移动设备列表的维护。模块所涉及的功能性的需求在表 3.1中列出，表中主要对列出的各功能性需求进行了内容、优先级方面的描述。

表 3.1: 设备管理模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R1	自动接入移动设备	用户将新的移动设备连接到设备管理服务器上，打开移动终端设备上的USB调试开关，设备管理服务自动读取设备相关信息将移动设备添加至设备列表。	高
R2	更新移动设备状态	用户选择某设备进行使用或使用结束或定时器检测到设备状态变化时，设备管理服务器会对设备占用情况进行监控，自动更新移动设备状态至上层服务器。	高
R3	查看移动设备列表	用户进入系统后，可以通过移动设备列表查看当前系统中所接入的移动设备信息。	中
R4	操控移动设备	用户在前端交互界面对设备进行操作后，设备管理服务器会将上层传输的操作指令发送至相对应的移动设备，完成设备的远程操控。	高
R5	选择移动设备	用户可以在系统的移动设备列表界面选择具体移动设备进行相关操控。	高

脚本录制模块包括Web前端与后端服务两个部分，主要负责用户操作监听、控件属性提取、用户单步操作记录、自动化测试脚本的开始及完成录制与脚本的生成功能。该模块通过Web前端向用户提供脚本录制相关功能，并进行相应处理后与设备管理模块交互完成用户操作从前端到设备终端的流程。同时脚本

录制模块实现了脚本的录制生成，为脚本跨平台回放提供了前提保障。该模块具体功能需求如表 3.2所示。

表 3.2: 脚本录制模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R6	监听用户操作	用户在前端交互界面进行设备操作，移动鼠标焦点到不同控件时获取该控件属性，对该控件周围以框线标明提示用户该控件已被选中。	中
R7	获取控件属性	用户在脚本录制过程中完成对某控件操作，后端获取具体控件，对控件截图、当前页面截图、页面XML属性进行获取。	高
R8	开始自动化脚本录制	用户在设备远程操作界面点击开始录制按钮，后端开始对用户在设备上的操作进行记录。	高
R9	记录用户单步操作	用户在开始录制脚本状态下在远程界面对设备进行操作，后端将用户操作及对应控件属性文件路径等记录到数据库。	高
R10	完成自动化脚本录制	用户在设备远程操作界面点击完成录制按钮，后端完成对用户操作的记录，更新数据库脚本状态、相关属性位置信息。	中
R11	生成自动化脚本	用户在点击完成录制按钮，后端将之前单步操作的记录文件进行统一打包，生成总的脚本文件压缩包，提供用户下载。	高

脚本回放模块包括Web前端、后端服务及图像处理服务三个部分，主要基于脚本录制过程生成的脚本提供在不同系统、型号移动设备上完成脚本的回放，该过程通过对脚本中控件截图借助图像处理服务实现新设备至上控件元素定位，完成跨平台、跨设备的脚本回放。为了便于用户管理录制脚本，该模块提供了脚本查看、删除功能。具体功能需求如表 3.3所示。

表 3.3: 脚本回放模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R12	开始自动化脚本回放	用户在脚本列表选择某个脚本，进入移动设备列表界面，选择某个回放设备，点击“开始回放”按钮，后端获取相关信息，根据脚本操作顺序在用户选择设备上开始回放。	高
R13	回放脚本单步操作	用户选择脚本、设备开始脚本回放过程，后端根据数据库提取到的脚本操作文件路径，按照相应顺序获取单步操作回放所需文件，处理、生成操控命令，完成单步操作回放。	高
R14	完成自动化脚本回放	用户选择脚本、设备开始脚本回放过程后，后端根据脚本记录对所有单步操作回放，完成后向前端反馈回放完成信息。	中
R15	切换其他设备	用户在设备远程操控界面点击切换设备按钮，进入移动设备列表界面，重新选择移动设备完成脚本录制或回放。	低
R16	查看自动化脚本列表	用户可以在脚本列表界面查看已经录制完成的自动化脚本的相关信息。	中
R17	删除自动化脚本记录	用户可以在脚本列表界面，点击某条脚本记录后删除按钮，删除该条脚本记录。	中

3.2.2　非功能性需求

跨平台脚本录制与回放系统非功能性需求列表如表 3.4所示，表中主要阐述了对系统预期达到的非功能性需求，对具体需求内容进行了阐述且给出了需求的优先级。本系统要求界面简洁，流程明确，用户能够在较短的时间内学习并使用系统完成脚本的录制与回放。同时考虑到系统可能与其他移动测试模块集成，不同模块进行独立开发，以微服务形式部署并对外提供服务，使系统能够保持良好的扩展性。为了应对图像处理技术的迭代更新以及扩展对其他移动系统平台设备的支持，本系统需要做好对接口的抽象设计，方便对图像算法的更新、替换及接入不同工具实现对不同平台设备的扩展与支持。在系统可维护性方面，系统应能够保证较长时间的稳定运行，在发生故障时能够通过脚本在5分钟内恢复正常状态。

表 3.4: 非功能性需求列表

需求编号	需求名称	需求内容	优先级
R18	易用性	系统流程明确，界面简洁明了，用户能够在短时间内能够熟悉并进行使用。	中
R19	可扩展性	系统可以增加其他移动测试相关模块功能，扩展成完整的移动自动化测试管理系统。 系统中图像处理以服务形式对外提供，能够实现图像匹配、布局匹配算法方便的更新、替换。 系统应该对设备管理流程内部阶段做好接口抽象，方便系统对不同平台移动设备实现支持。	中
R20	可维护性	系统可以在较长时间能维持稳定运行的状态，在系统出现故障后重启系统能够保证5分钟内重新恢复运行状态。	高

3.3　系统用例图

本文设计实现的跨平台脚本录制与回放系统具体用例图如图 3.2所示。系统主要由测试人员与计时器两个角色组成。定时器主要涉及移动设备状态的更新，定时器会定时检查与系统连接的移动设备的状态，保证系统中记录的设备状态与真实设备状态的实时同步。

测试人员是系统的主要角色，涉及设备、脚本录制与回放相关功能。首先测试人员可以接入新的移动设备到系统，可以通过设备列表查看当前系统中接入设备信息，可以选择具体移动设备并对移动设备进行远程操控，在与设备交互的过程中同时涉及到了移动设备状态的变换。其次测试人员可以进行脚本录制与对录制完成的脚本进行跨设备或跨平台回放，而录制与回放过程都涉及到了对于移动设备的相关操作，因而用例呈现出包含关系。脚本回放时测试人员

可以快速进行设备间的切换，进行多平台、多设备脚本验证。最后测试人员可以对已录制完成的脚本记录进行查看与删除。

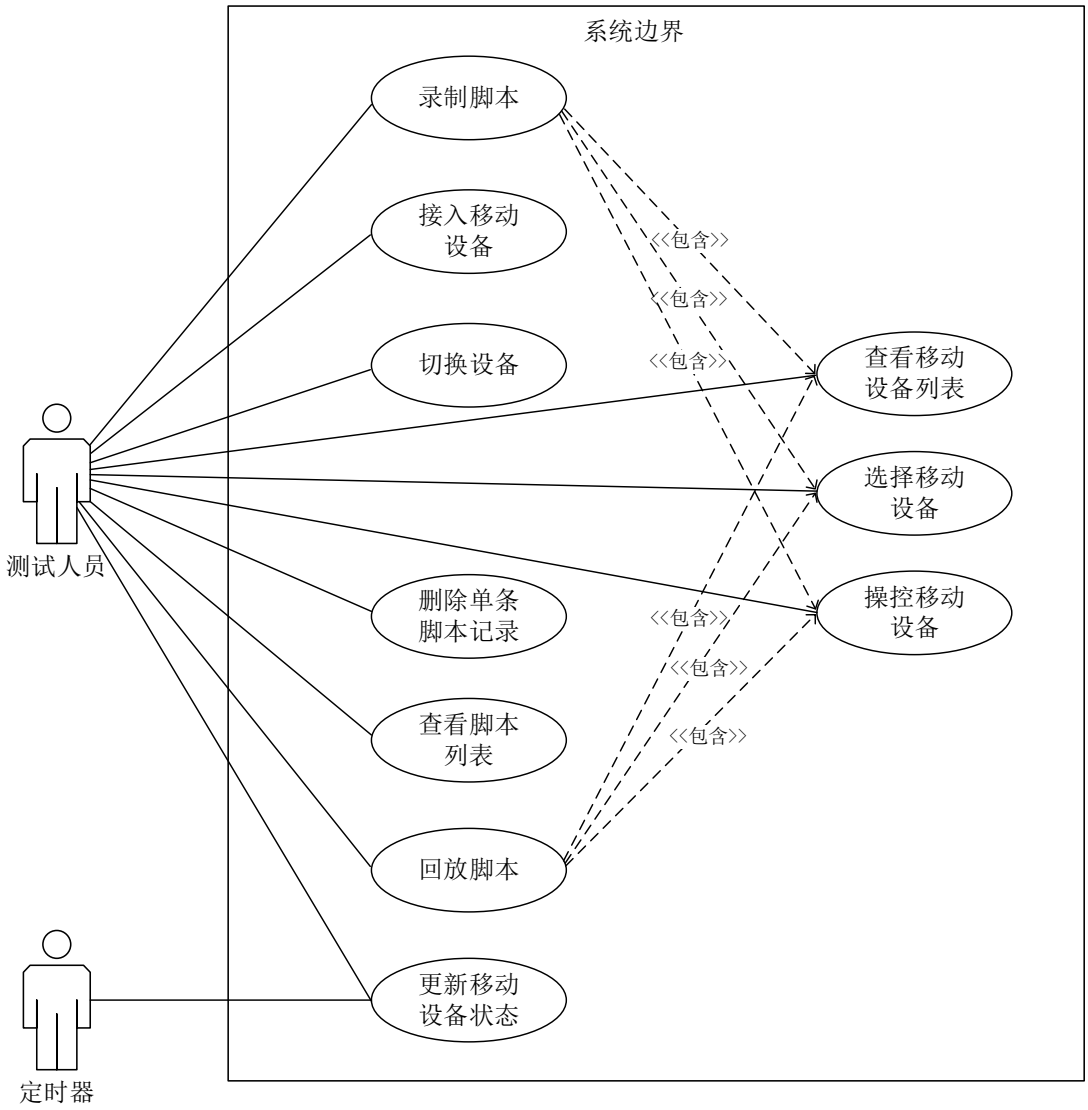

图 3.2: 系统用例图

如表 3.5所示，系统共涉及10个主要系统用例，分别为接入移动设备、更新移动设备状态、查看移动设备列表、操控移动设备、选择移动设备、录制自动化脚本、回放自动化脚本、切换设备、查看脚本记录列表、删除脚本记录。表中对各用例进行编号，同时给出用例与功能需求之间对应关系，表的最后一列给出对应的功能需求编号。

表 3.5: 系统用例表

用例编号	用例名称	功能需求编号
UC1	接入移动设备	R1
UC2	更新移动设备状态	R2
UC3	查看移动设备列表	R3
UC4	操控移动设备	R4
UC5	选择移动设备	R5
UC6	录制自动化脚本	R6、R7、R8、R9、R10、R11
UC7	回放自动化脚本	R7、R12、R13、R14
UC8	切换设备	R15
UC9	查看脚本记录列表	R16
UC10	删除脚本记录	R17

以下通过用例表对给出的系统用例进行详细描述。

接入移动设备是系统对设备进行管理的入口，测试人员将新的移动设备通过USB手动连接到服务器，系统针对不同系统（目前支持移动设备系统为Android及iOS）设备通过不同工具获取设备信息更新内存中设备列表并向脚本录制回放服务进行同步。用例描述如表 3.6所示。

表 3.6: 接入移动设备用例表

描述项	说明
用例编号	UC1
用例名称	接入移动设备
参与者	测试人员
用例描述	新的移动设备连接到系统，完成设备系统接入，能够通过系统使用设备
前置条件	系统处于正常运行状态
正常流程	1. 测试人员将新的移动设备通过USB形式连接到系统服务器，确认移动设备允许接入当前服务器 2. 设备管理服务自动获取设备信息，更新设备管理服务在内存中维护的设备列表，将新设备添加至设备列表中
后置条件	新接入的移动设备可以被测试人员选择使用
异常流程	2. 测试人员未设置设备允许当前服务器接入，设备管理服务不能获取设备信息，不能将新设备信息正常更新到设备列表

更新移动设备状态主要是对系统中多服务间保存的设备状态进行同步维护，保证在测试人员在使用设备时触发设备状态变更或定时器检测到设备状态改变时，设备管理服务自动更新系统中设备状态并保证系统中维护的设备状态信息的同步一致性。用例描述如表 3.7所示。

表 3.7: 更新移动设备状态用例表

描述项	说明
用例编号	UC2
用例名称	更新移动设备状态
参与者	测试人员、定时器
用例描述	在测试人员操作设备或定时器定时检查设备状态，当设备状态发生改变时，自动更新系统中设备状态记录，与真实设备状态保持一致
前置条件	测试人员操作设备或其他因素导致设备状态更改
正常流程	1. 测试人员选择或完成使用设备；定时器定时检查设备状态，向系统同步设备状态 2. 系统比对当前维护状态与接收设备真实状态，设备状态不同时进行状态更新
后置条件	系统维护的设备的状态保持与真实设备状态相同
异常流程	无

查看移动设备功能目的是对测试人员展示当前系统中设备基本信息及对应状态。选择移动设备功能是通过查看移动设备从中选择空闲设备跳转至设备操控界面，建立与设备之间连接，方便测试人员进行脚本录制或脚本回放。用例描述如表 3.8所示。

表 3.8: 查看/选择移动设备用例表

描述项	说明
用例编号	UC3、UC5
用例名称	查看/选择移动设备
参与者	测试人员
用例描述	测试人员通过设备列表可以查看当前系统中接入的所有移动设备，可以选择其中一款空闲设备进行操控
前置条件	测试人员点击查看设备列表
正常流程	1. 测试人员点击查看设备列表 2. 显示当前系统设备列表 3. 测试人员选择某一空闲设备 4. 进入设备操控界面，初始化加载设备
后置条件	测试人员能够查看当前系统中设备情况，选择某空闲设备后可以对该设备进行远程操控
异常流程	4. 设备初始化加载失败，返回设备列表让测试人员重新选择设备

操控移动设备功能主要是通过对Web端远程界面控件点击实现对真实设备同步操控。该功能为系统实现脚本录制的核心功能，实现了测试人员与设备的远程交互，保留了真机测试时操控方式，方便测试人员快速了解、使用系统。用例描述如表 3.9所示。

表 3.9: 操控移动设备用例表

描述项	说明
用例编号	UC4
用例名称	操控移动设备
参与者	测试人员
用例描述	测试人员在设备远程同步界面，可以点击界面控件元素，对真实设备进行操控
前置条件	测试人员进入设备操控界面，设备初始化加载完成
正常流程	1. 测试人员进入设备操控界面，设备初始化加载完成 2. 测试人员点击界面控件元素，系统监控到设备操控事件，将操控转换为设备可运行的指令传递于设备管理服务器，最后传达到设备完成相应操作 3. 设备界面同步传输至前端页面，展示操控过程
后置条件	测试人员可以远程操控真实设备
异常流程	2. 设备因异常中断连接，重新对设备进行初始化加载

录制自动化脚本功能是本系统核心功能之一。基于查看、选择、操控移动设备功能，建立与移动设备的远程连接。测试人员点击脚本录制按钮，Web 端即对界面所有操作进行监听并向后端发送操控信息，后端服务执行对单步操作的记录流程，同时将操作顺序记录至脚本，直至测试人员点击录制完成按钮停止操作监控及后端服务记录，完成本次脚本录制流程。对脚本单步操作记录主要涉及控件及当前界面截图、XML文件的提取。用例描述如表 3.10所示。

表 3.10: 录制自动化脚本用例表

描述项	说明
用例编号	UC6
用例名称	录制自动化脚本
参与者	测试人员
用例描述	测试人员在设备操控界面，点击脚本录制按钮，开始按照测试用例流程对设备进行操作，完成所有操作后，点击录制完成按钮完成脚本录制
前置条件	测试人员进入设备操控界面
正常流程	1. 测试人员进入设备操控界面，设备初始化加载完成，点击脚本录制按钮 2. 测试人员按照测试用例流程开始分步操控移动设备 3. 后端对测试人员操作进行单步记录，获取控件截图、界面截图、XML等信息，同时将脚本文件位置、设备等信息写入数据库，将操作转化为相应的可执行命令传递至移动设备完成设备操控 4. 测试人员按照测试用例流程完成操作，点击完成录制按钮 5. 后端更新数据库脚本记录信息，同时将所有操作步骤截图、XML等文件Zip压缩生成脚本文件提供Web端下载
后置条件	测试人员能够查看、删除录制的脚本，且能够选择录制的脚本在不同的设备上完成脚本的回放
异常流程	无

回放自动化脚本功能实现了对系统中录制脚本在不同移动设备上进行回放，以验证不同机型、不同平台上移动应用程序的功能是否正常。测试人员通过选择某具体录制脚本与回放设备，在脚本回放界面点击回放按钮，系统通过脚本记录编号从数据库中获取对应脚本记录信息，对脚本文件中记录控件截图与当前设备界面截图进行处理，通过调用图像处理服务获取当前步骤操作控件在新设备上真实坐标，接着将坐标转化为设备可执行命令完成对设备的操控达到操作回放的效果。为方便测试人员及时查看脚本执行情况，脚本回放界面与移动设备界面实时同步，可视化脚本执行过程。用例描述如表 3.11 所示。

表 3.11: 回放自动化脚本用例表

描述项	说明
用例编号	UC7
用例名称	回放自动化脚本
参与者	测试人员
用例描述	测试人员在脚本记录列表可以选择某脚本记录，选择需要回放的移动设备，可以在不同机型、不同平台移动设备上完成录制脚本的回放
前置条件	测试人员进入脚本记录列表界面，且系统中存在已录制完成的脚本记录
正常流程	1. 测试人员进入脚本记录列表界面，选择某项脚本记录，点击回放按钮进入系统设备列表 2. 测试人员选择想要回放的具体设备，点击开始回放，进入设备操控界面，初始化加载设备 3. 系统获取脚本执行信息，按顺序对脚本文件提取，控件定位算法对分步操作控件对象在新设备上进行定位 4. 获取控件位置坐标后生成不同平台下移动设备操控命令，传送至具体设备，完成相关操作回放 5. 脚本所有操作完成回放后，反馈脚本回放完成信息至Web端
后置条件	测试人员能够验证该款应用脚本录制功能在选定设备上能否正常运行
异常流程	3a. 对当前步骤操作控件采用图像匹配方式进行定位 3b. 当前步骤操作控件图像匹配方法不能正常获取定位，采用布局匹配方法，获取控件位置坐标

切换设备功能的目的是为测试人员在脚本回放时能够便捷、快速的实现对当前操控设备的切换，能够使测试脚本在不同移动设备上进行回放，方便对移动应用在不同机型或不同平台设备上进行回归测试验证。测试人员在对当前设备完成测试后无需返回脚本记录列表重新选择脚本记录及回放设备，点击页面切换设备按钮，弹出设备列表即可选择不同设备，重新建立Web端与设备之间的连接完成设备切换，以提高应用程序在不同设备间进行测试的效率。用例描述如表 3.12 所示。

表 3.12: 切换设备用例表

描述项	说明
用例编号	UC8
用例名称	切换设备
参与者	测试人员
用例描述	测试人员在设备操控页面，点击切换设备按钮可以查看当前设备列表，选择不同设备后重新对新设备进行初始化加载
前置条件	测试人员进入设备操控界面
正常流程	1. 测试人员在设备操控界面点击切换设备按钮 2. 界面弹出当前系统设备列表，测试人员选择切换至的移动设备 3. 设备操控界面窗口重新初始化加载新设备
后置条件	测试人员可以对新切换的设备进行相关的操控
异常流程	3. 新设备初始化加载失败，弹出当前系统设备列表让测试人员重新选择移动设备

查看脚本记录主要对系统中脚本录制设备、时间及相关描述信息进行展示，测试人员可以通过脚本记录列表查看并选择测试脚本用于回放测试。删除脚本记录是对录制完成的脚本记录提供删除功能，避免系统中录制脚本的冗余。测试人员可以点击脚本记录对应的删除按钮，后端服务会对测试记录相应状态进行修改，同时被删除脚本记录在记录表中不可见。用例描述如表 3.13所示。

表 3.13: 查看/删除脚本记录用例表

描述项	说明
用例编号	UC9、UC10
用例名称	查看/删除脚本记录
参与者	测试人员
用例描述	测试人员通过脚本记录列表可以查看已录制完成的脚本，可以对列表中记录进行删除操作
前置条件	测试人员点击查看脚本列表
正常流程	1. 测试人员点击查看脚本记录列表 2. 显示当前系统中录制完成的脚本记录 3. 测试人员点击某条记录后删除按钮 4. 系统对当前记录状态进行修改，被删除脚本记录从当前列表中移除
后置条件	测试人员能够查看系统中录制完成的脚本记录，对单条脚本记录进行删除
异常流程	无

3.4　系统总体设计与模块设计

3.4.1　系统总体设计

本系统主要由用户交互前端服务，脚本录制回放、设备管理及图像处理几个大的模块组成，系统总体的部署情况如图 3.3所示。其中脚本与录制回放服务

器为系统的中心，连接Web前端、设备管理服务器及图像处理服务器。为了保证移动终端设备界面到前端页面显示的流畅性与稳定性，脚本录制回放服务器与设备管理服务器、前端均采用Netty框架，以TCP或WebSocket实现信息的传输。图像处理服务器方面采用RESTful接口形式对外提供服务，服务独立开发与部署，使得图像处理算法进行改进、变动时对整个系统无感知化。

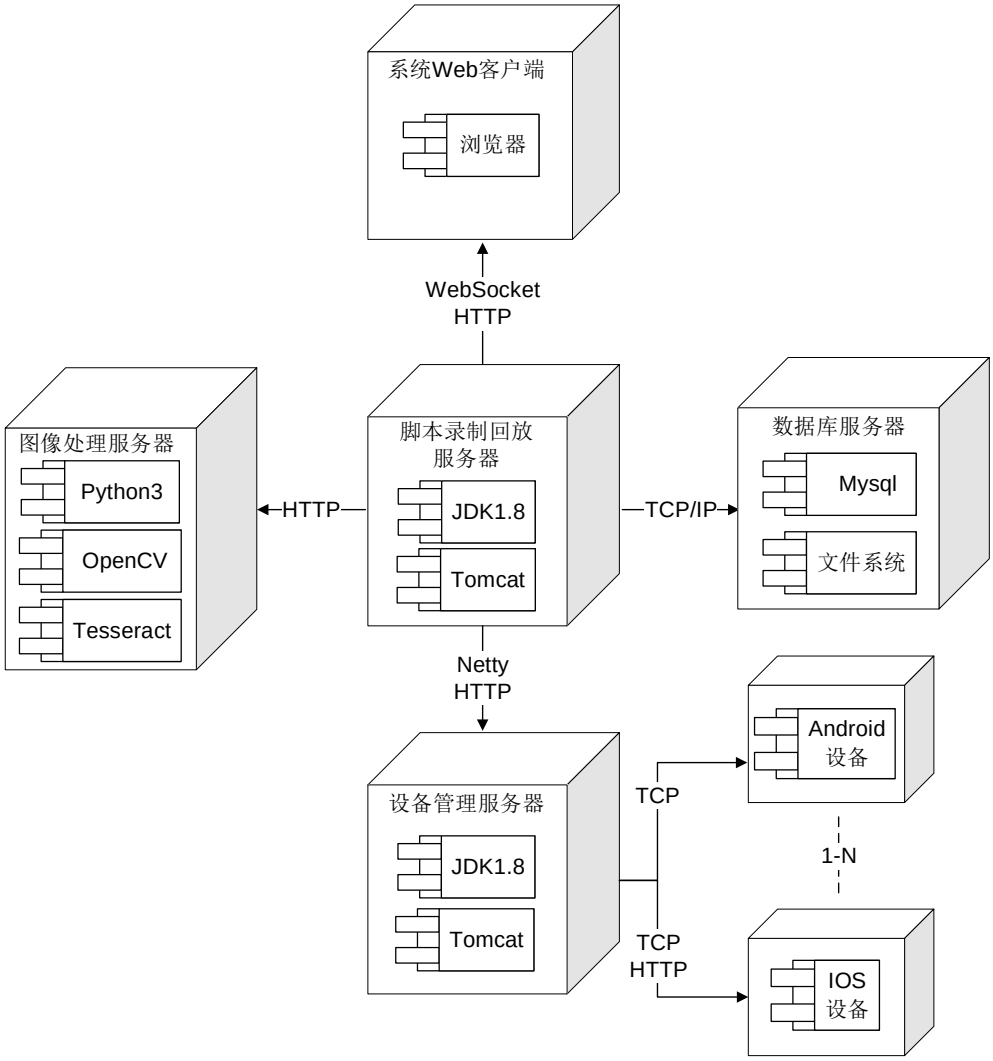

图 3.3: 脚本录制与回放系统总体部署图

图 3.4给出了整个脚本录制与回放系统的总体架构，给出了部署图中各个服务器所涉及的主要功能服务。下面对各个服务器所涉及的具体的功能、业务等进行简要的介绍。

脚本录制与回放系统Web管理端直接与测试人员进行交互，主要涉及设备管理、脚本管理、脚本录制回放三个部分，能够响应用户的查看、选择、操控

设备，录制、回放、查看、删除脚本等操作。其中涉及到设备界面同步传输、操作指令传输的信息采用Netty框架与WebSocket协议组合的方式完成，其余常规操作仍以HTTP交互方式实现。

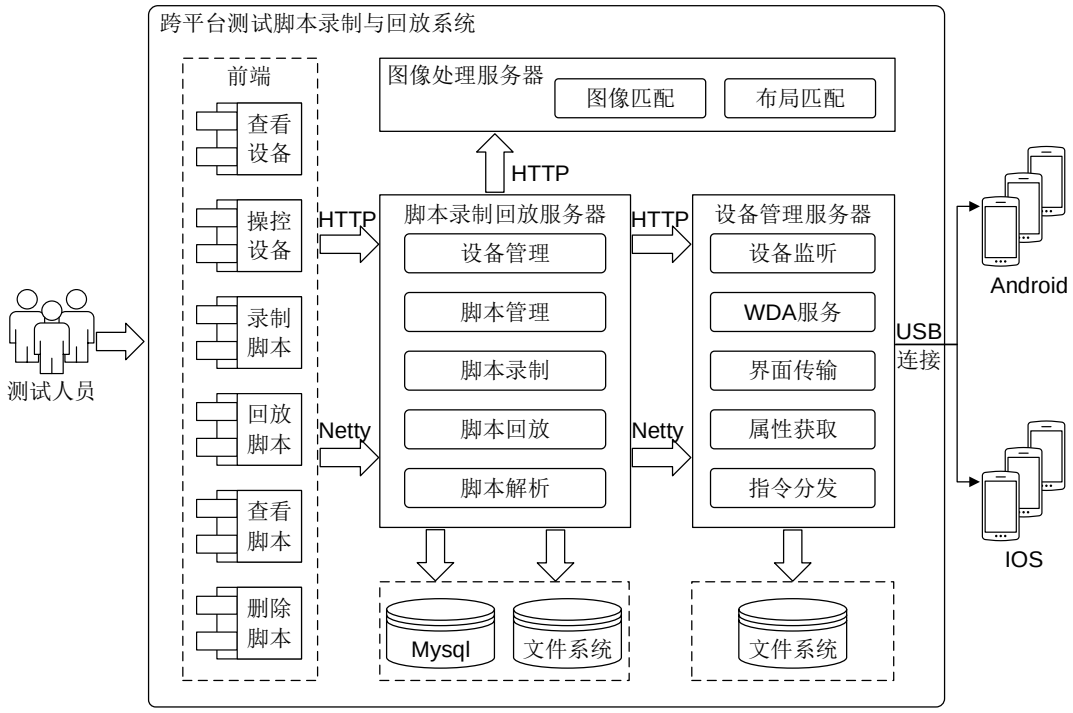

图 3.4: 系统总体架构图

脚本录制回放服务器是系统的主要业务处理服务器，作为底层设备管理服务器与上层Web管理端的中转，接收底层服务器获取的设备界面信息中转至上层Web管理界面实时展示给用户，同时将上层捕获的用户操作信息转至底层进行处理，进而转化为设备可执行的终端命令，完成测试人员到移动设备的远程交互流程。此外该层提供了对设备列表的维护、设备状态维护，便于用户能够获取到系统中所有移动设备及状态信息。同时提供了脚本管理服务，包括脚本列表的维护、脚本记录删除功能。

脚本录制与脚本回放为该层主要业务部分。脚本录制在接收到开始录制命令后，开始对用户操作信息进行处理，新增脚本记录到数据库中，同时对每步操作进行控件截图、设备当前页面截图、XML文件提取等操作，将上述文件存储至文件系统中记录路径到数据库，对每一步用户操作重复上述过程，直至接收到录制完成命令后将所有步骤文件打包处理作为脚本文件提供用户下载查看。脚本回放在接收开始回放命令后，通过脚本记录编号得到数据库中记录的相关脚本文件路径及脚本操作相关信息，将该脚本分解为单个步骤依次进行回

放，单步操作回放过程中调用图像处理服务，首先利用图像匹配技术对控件在新设备上定位，在服务识别异常时再通过布局匹配技术获取定位信息，最终将获取到的控件坐标信息传递于底层，由下层服务器转化为相应设备操作命令完成操作的回放，循环执行上述过程完成脚本回放。具体模块设计描述在下一小节中给出。

设备管理服务器主要针对底层移动设备的一系列管理，通过USB与移动设备直接连接，包括对设备的接入、状态监控、界面获取、指令分发、执行等。针对Android、iOS不同平台设备分别选取ADB、WDA工具实现了对应平台设备的操控。通过Minicap工具实现设备界面的远程同步。脚本录制回放服务可以通过该层封装的接口实现对移动设备的操控、信息获取。设备管理服务单独开发，通过接口调用实现通讯，保证系统模块间“高内聚、低耦合”，同时提高了系统的可扩展性。

图像处理服务器是系统的关键优化点，也是脚本文件到不同设备界面坐标转化的实现所在。图像处理服务以RESTful服务形式进行单独封装部署，即保证了算法在优化、更新时仅需保证对外接口的完整性，保证了系统的可维护性、可修改性。该部分通过其他业务流程构建的图形化控件属性描述文件及新设备当前页面截图作为输入参数，由图像匹配、布局匹配两种方式来保证获取控件在新设备页面定位，并将具体坐标进行返回，忽略了平台、设备等特性，进而保证了脚本的跨平台、跨设备特性。

3.4.2　设备管理模块设计

设备管理模块主要用于设备的接入、设备状态的监听、设备数据的传输及设备的查询，图 3.5对整个模块的运行流程进行了描述。

在系统启动时设备监听器与定时器会随之启动，设备监听器在新设备接入系统时会触发接入事件，随后系统对新接入的移动设备获取Udid、设备名称、RAM大小、操作系统等信息，新增设备记录到设备列表中，将该设备状态至空闲。同时监听器对系统中设备状态进行监听，当设备状态由于设备调用而在空闲状态与占用状态进行迁移时，对系统中维护的设备列表进行相应的修改，设备列表以Map形式存储于内存。

设备状态检查定时器以每2秒的频率查询系统中维护的设备列表，获取当前设备的信息，同时将设备的状态信息向上层服务器同步，维护设备在整个系统中状态的一致性。

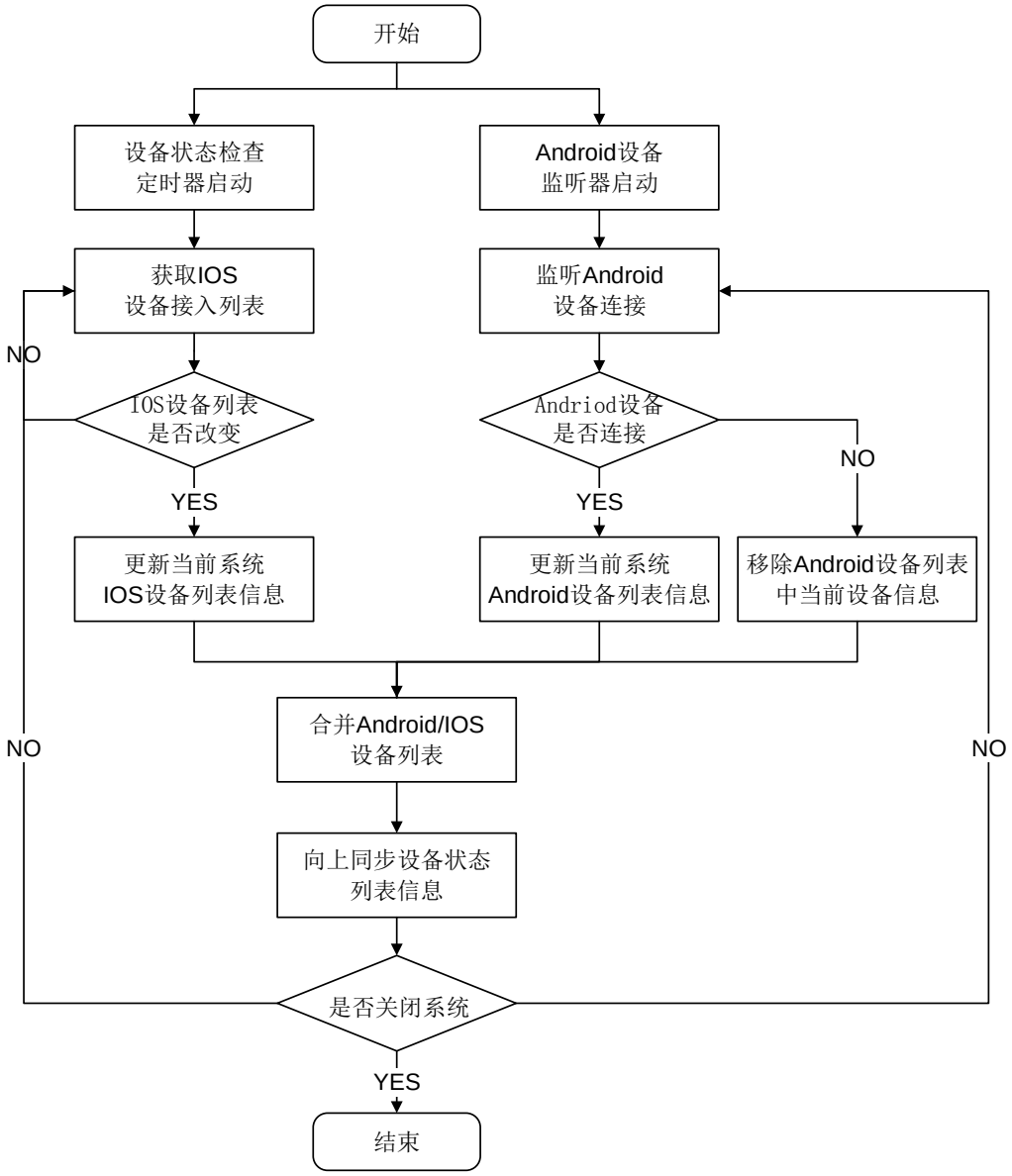

图 3.5: 设备管理模块流程图

同时该模块与移动设备硬件直接连接，是从设备中获取必要信息、界面信息及驱动设备按照指令运行的基础。因此该模块提供了设备指令分发至具体设备、设备界面数据向上传输的功能。该模块通过MiniCap 与设备建立连接，获取设备界面信息通过Netty框架进行TCP传输，将真机页面实时传输至远端。且该服务接收来自上层服务器的操控指令，通过ADB或WDA工具转化为在不同平台下可执行的设备指令，根据不同设备的Udid分发到具体的设备，完成对设备的远程操控。除了对应用设备的常规操作，该模块也对获取设备当前页面截图、获取当前页面XML文件，解析XML文件提取节点元素等进行了封装，更好

的为上层服务提供设备相关信息。

3.4.3　脚本录制模块设计

脚本录制模块是脚本录制回放系统的核心模块之一。该模块主要为Web端操作界面与底层设备管理模块交互的中转层模块，同时提供脚本信息记录，更新，及构建脚本回放时需要的图像化控件属性文件的记录等。主要流程如图 3.6 所示，大致包括设备远程连接、单步操作记录、生成脚本等过程。

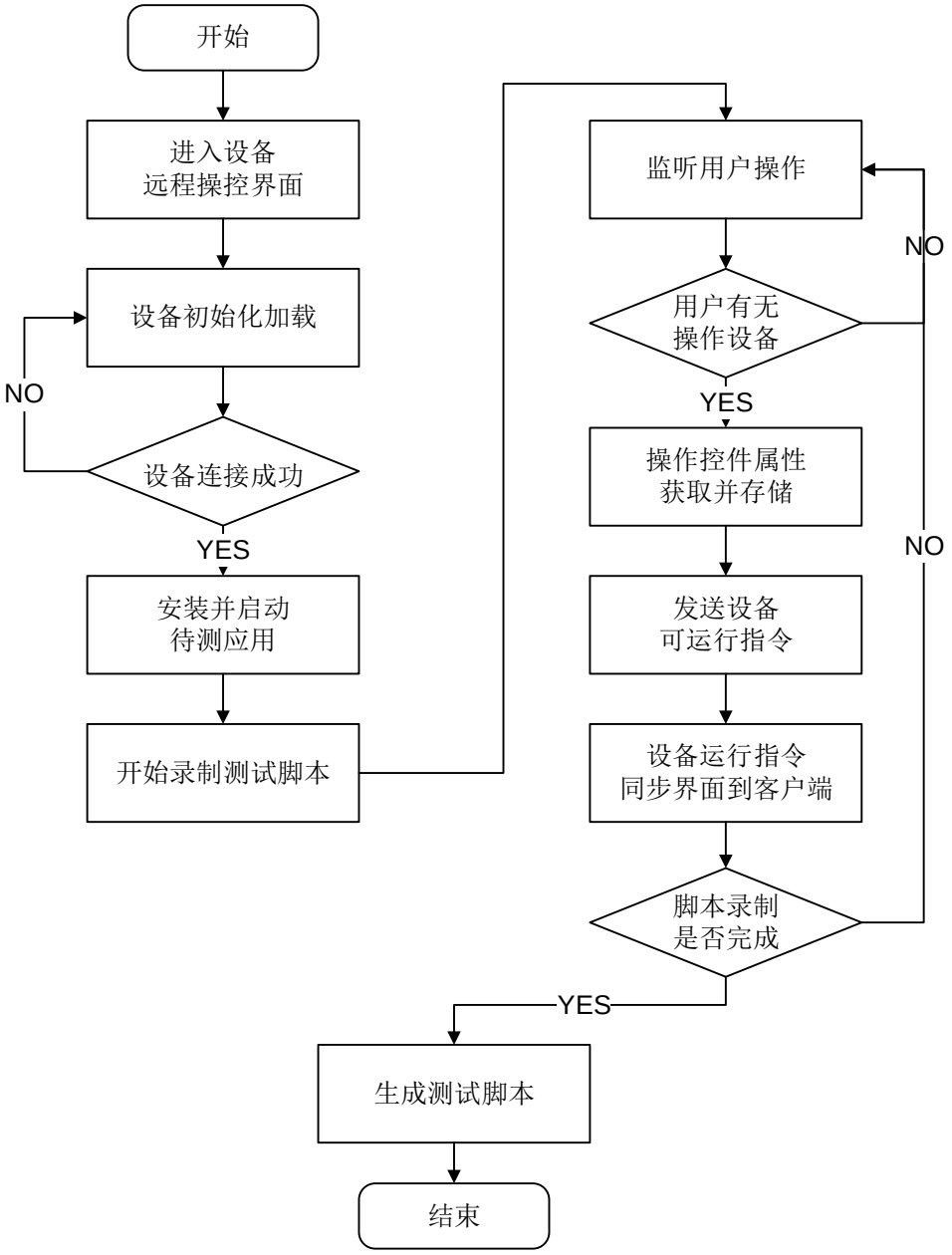

图 3.6: 脚本录制模块流程图

设备远程连接与远程设备操控是实现脚本录制的基础，该过程是在开始录制脚本之前完成的。首先当测试人员选定操控设备即进入远程设备操控界面，Web前端与该层建立WebSocket连接，通过Netty框架实现TCP传输，同时该层获取前端选择的设备Udid向设备管理服务器获取该设备页面数据，该过程的实现同样基于TCP进行传输。双层Netty框架下的TCP连接设计，即降低了脚本录制回放业务层与设备操控层间耦合性，增加了系统的可扩展性，又保证了界面图像数据量较大的情况下数据传输的实时性。

脚本的录制是对单步操作记录的一个循环过程，完成对单步操作的录制仅需对操作顺序进行记录即可完成脚本的记录。单步操作记录主要是对操控控件属性进行提取记录。所提取的控件属性包括传统属性如Id、XPath、文本等，在此基础上通过前端传来的XML控件节点信息，对当前页面XML 文件、当前页面截图及控件截图进行提取。将提取到的文件按照文件夹嵌套的形式存储，根文件夹代表当前脚本，每个子文件夹为对每次操作所提取出的控件属性文件，同时会在根目录下生成操作执行顺序文件。

脚本的生成是在测试人员确认脚本录制完成后触发。对所有操作提取到的文件并进行打包，返回文件路径存入数据库，提供测试人员下载查看。本系统所生成的脚本文件非传统编码意义上脚本文件，不能支持其他框架下回放。

3.4.4　脚本回放模块设计

脚本回放是基于录制过程中生成的脚本文件，对其在不同平台、机型设备上回放操作的过程。该模块主要流程如图 3.7所示，主要包括设备远程连接、脚本解析，及控件的定位。其中控件定位基于图像匹配与布局匹配两种方式实现，保证脚本的稳定性。

设备远程连接与录制过程相同，此处不再重复说明。当测试人员选定脚本与回放机型后，该模块会从数据库中获取脚本存储路径，对脚本文件解压处理，对所得的脚本文件夹按照执行顺序分别进行单步处理。单步脚本回放过程，获取新设备当前页面截图，首先以图像匹配进行控件截图与页面截图匹配，匹配成功可获取控件在页面中位置信息，视为定位成功，否则对页面截图进行布局划分处理，根据控件布局位置坐标找寻截图布局中相应模块，查找成功则为定位成功，否则认为该脚本回放异常，提示测试人员检查并结束回放。匹配到的位置信息会以中心坐标形式反馈，传递至底层设备管理模块转化为可执行命令完成设备端的回放操作。之后依次执行所有步骤，完成后提示脚本回放成功。

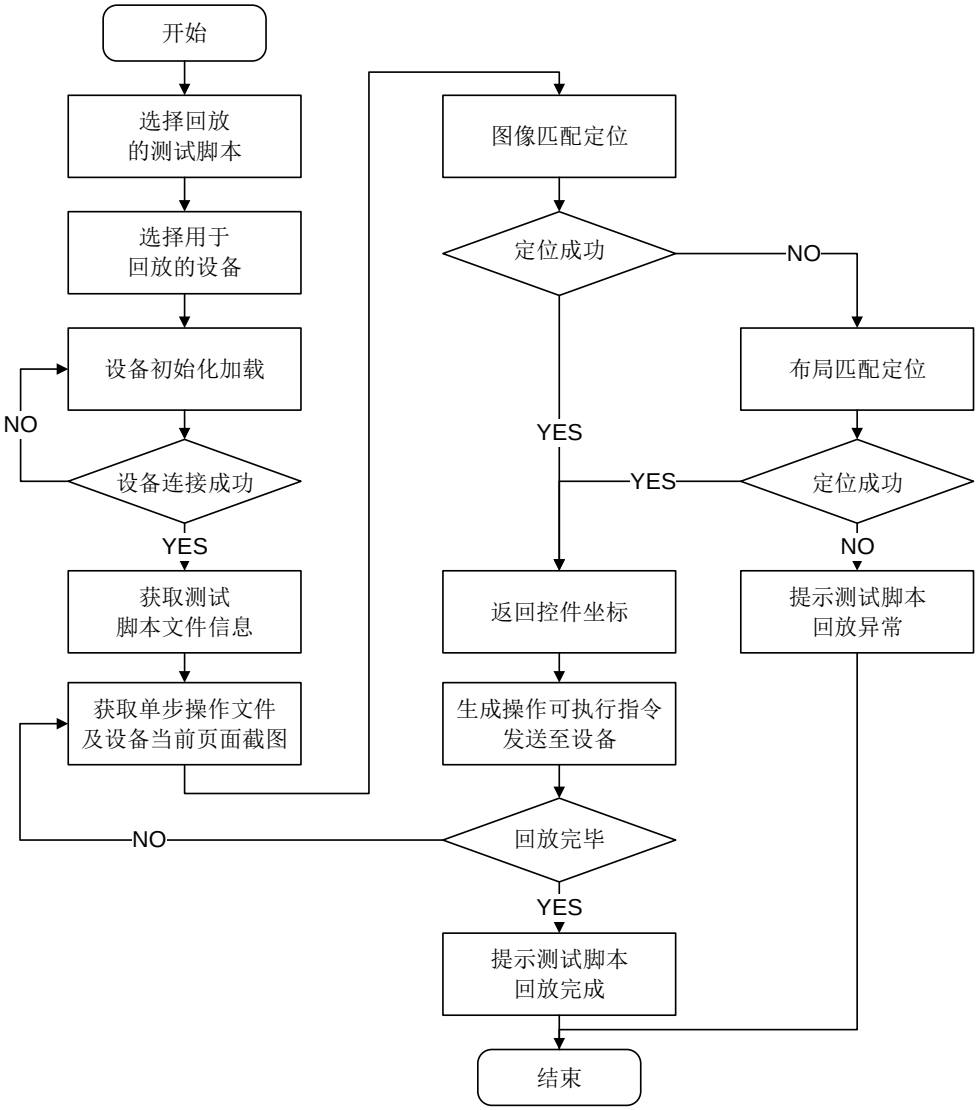

图 3.7: 脚本回放模块流程图

3.5　测试脚本回放实现方案

为了实现录制脚本的跨设备、跨平台与支持Hybrid等的回放，系统中利用录制过程中提取到的控件、页面图像信息，传统控件属性等利用图像处理技术对控件在新设备上进行定位 [38]。系统提供了图像匹配与布局匹配两种控件定位策略，保证在新设备上能够实现对控件的定位。

3.5.1　图像匹配定位

图像匹配定位是指通过脚本中每步操作所记录的“操作控件截图”与新设备“当前页面截图”之间采用图像特征匹配算法，实现对控件在页面上的位置

定位。该种方案局限在于当新设备页面中控件图像发生较大变动或当页面中存在多个相同的图像导致匹配至多个区域时，无法对控件进行正常定位。

图像特征匹配算法以目标图像与待匹配图像作为输入，将最终于待匹配图像上匹配到的区域顶点坐标值作为输出结果。算法主要包括预处理、特征提取、特征匹配、消除错配、计算畸变五个主要的过程，图像匹配算法的流程如图 3.8所示。该算法具体设计与每个过程说明如下。

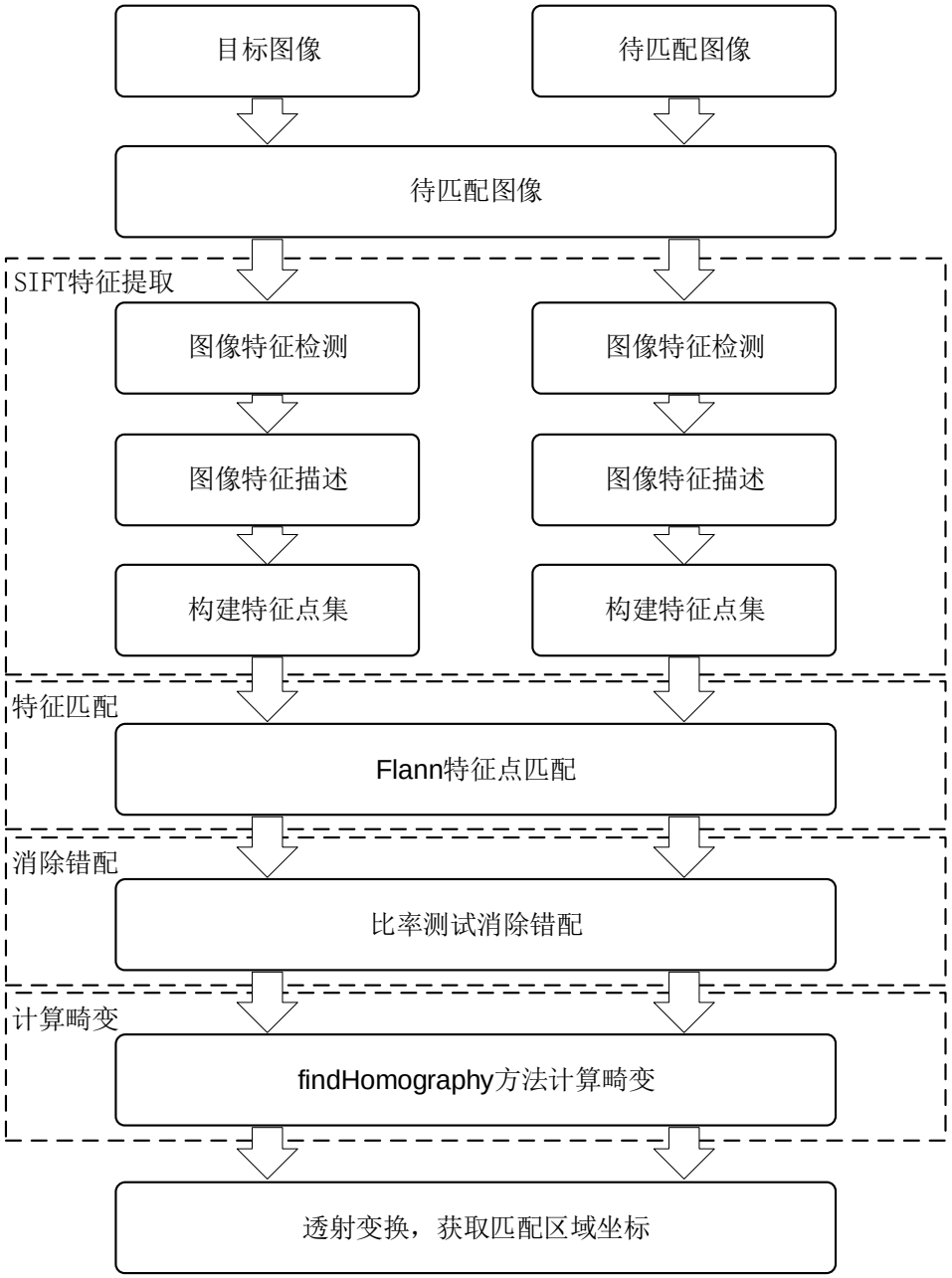

图 3.8: 图像匹配算法流程图

预处理过程。对输入图像进行灰度变换处理。由于后面流程中图像的颜色等信息不作为处理属性，将图像转为灰度图提高了图像的质量，增加了图像的清晰度、对比度，使得在后续处理时能够得到更好的效果。

SIFT特征提取。该过程主要应用SIFT特征提取算法，该过程包括图像特征的检测、图像特征描述符与特征点集的构建。OpenCV中SIFT已对上述过程进行了封装，仅需对xfeatures2d.SIFT_create()创建的SIFT对象通过detectAndCompute()方法即可得到图像的特征点集与描述子集。分别对目标图像与待匹配图像进行处理，得到两组特征点(特征点集分别以K_{target},K_{source} 表示)与描述子集合(描述子集分别以D_{target},D_{source} 表示)。

特征匹配。对于特征点的匹配是通过FLANN 库完成的，FLANN是快速估计最近邻的库，包含了一些为大数据集内搜索快速近邻的优化算法，相比Brute-Force匹配，FLANN 在处理大数据集时具有更快的速度。在对两个特征点集合的处理中，采用KD-Tree索引，KNN算法找到D_{target} 中与D_{source} 距离最近的两个点作为匹配点，完成对特征点的初步匹配。

消除错配。由KNN筛选过的匹配点可能出现错误匹配，因此对匹配点集进行误配筛选十分必要。SIFT算法提出者Lowe给出的比率测试方法能够较好的解决这个问题。比率测试即对上一过程中每个特征点提取到的两个最近邻特征点，按照公式ratio $=\frac{D_{\min}}{D_{\mathrm{sec}\,ond_\min}}<\delta$计算比值，当该值小于某一阈值时，即认为最近点匹配较好，否则认为该匹配为错误匹配，并从集合中剔除 [29]。经验表明取值为0.5时能够避免90%以上的错误匹配。

计算畸变。由于目标图像往往在待匹配图像中会存在旋转、放缩等畸变，为了更为精确的得到匹配区域的位置，采用findHomography方法计算两者之间的单应矩阵，最后对目标图像进行透视变换得到匹配区域的位置坐标信息。

图像匹配处理能够较快的完成对控件截图到新设备页面图像的区域匹配，几乎能够完成从图像输入到坐标返回的毫秒级处理，很大程度上保证了回放过程中的步骤之间的流畅性。

3.5.2　布局匹配定位

布局匹配则是对脚本文件中获取到的页面截图图像分割、OCR等技术进行布局刻画，获取操控控件在当前页面布局下的位置坐标信息。同时对新设备当前页面截图进行相同布局刻画操作，按照提取出的坐标信息对控件进行定位。应用同一方法对布局相同的界面进行处理，能够保证前后结果的统一性。该方案采用界面布局信息作为回放依据，由于大多应用在更新中整体页面布局及相

应控件功能均不会出现较大变动，且布局信息忽略了图像内容、文字等特征，能够更好的实现跨平台、跨设备的应用。

布局匹配以REMAUI工具[1] 为基础，该工具主要实现功能为通过界面图片逆向生成页面代码。其中对页面布局刻画部分是本文使用并改进为布局匹配方法的基础。工具分别以图像分割技术Canny 与OCR对图像页面布局进行刻画，结合两种方式处理结果生成页面布局。界面被分行、分列分割，并对每个元素给出布局二维坐标用于控件的定位。具体处理流程图如 3.9所示。

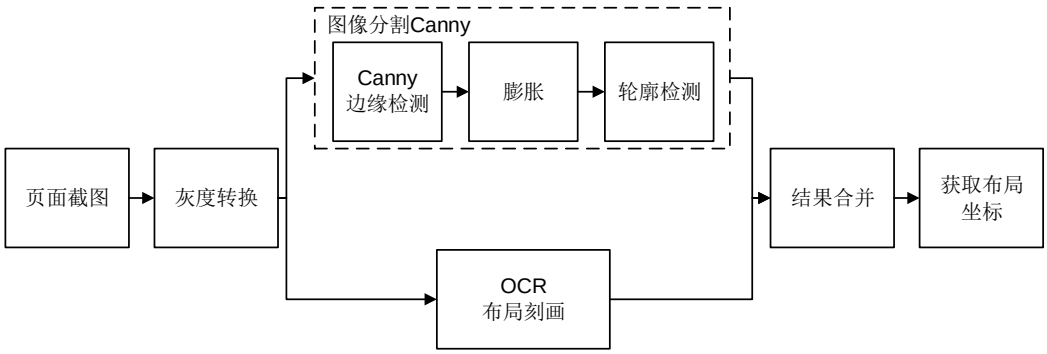

图 3.9: 布局匹配算法流程图

图像分割部分采用了Canny算法对灰度化的页面截图进行快速的边缘提取，由于Canny 算法对于边缘的刻画较为精细，如果立刻对结果进行轮廓检测，会导致出现过多的细小冗杂的边缘轮廓，而这些细小的轮廓在布局刻画方面往往没有意义，反而会影响对轮廓数据的处理。因此通过对图像边缘膨胀处理，将冗余、细小的轮廓联通，保留较大的，有意义的控件、文本等轮廓。轮廓检测通过findContours函数实现，提取轮廓的完整层次结构以矩形边框四个顶点坐标进行存储。

OCR部分通过Tsseract-OCR引擎实现对图像word、line、block三种精度的分割。通过定义的规则剔除长或宽过小、过长等不符合应用中真实场景的边框。

最后通过OCR结果对Canny检测到的轮廓结果进行轮廓上的合并，提取出与页面控件较为拟合的轮廓集合，但该集合仍存在很多冗余。通过对不同应用页面的多次试验总结，对得到的数据通过以下规则进行筛选基本能够完成对页面的布局层次刻画。

(1) 清除轮廓中长、宽小于阈值的轮廓，试验验证当阈值为当前设备屏幕宽度的2% 时，该轮廓元素不为功能性控件轮廓。

[1]https://github.com/soumikmohianuta/pixtoapp

(2) 清除轮廓的内部轮廓，当该轮廓长或宽超过对应设备宽度的60%以上时，忽略此条。应用一般在控件占据页面较小区域时，内部控件功能一般等同于自身，因此对于此类内部轮廓没有意义。

对筛选过的轮廓集中轮廓按照自上由下分层，并对同层轮廓自左到右编号，所得二维坐标即为布局匹配中构建的控件位置坐标。录制过程中通过控件位置所落入的轮廓获取该位置坐标，回放时按照位置坐标进行轮廓的查找，以实现通过布局的效果完成控件在新设备上的定位。

3.6　数据库设计

如图 3.10所示为脚本录制与回放系统的数据库设计，主要包含以下实体：用户、应用、设备、脚本记录，关系包括上传应用与录制脚本。

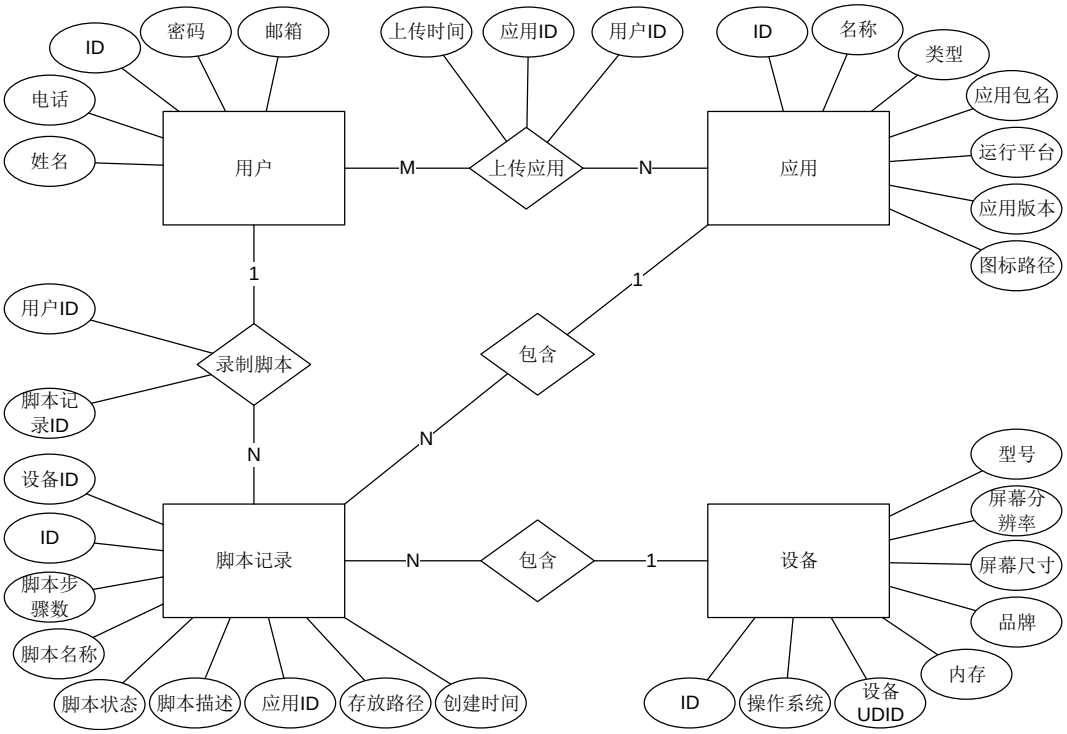

图 3.10: 系统数据库关键实体关系图

用户即指测试人员，每个测试人员可以通过系统上传多个需要测试的应用App，同时同款App也可以被多个不同的测试人员上传，因此用户与应用之间为多对多的关系，将上传应用关系表单独抽离，可以使得应用表中仅关注于应用相关信息，减少用户与应用上传关系带来的冗余数据。同时测试人员可以在系统中录制多个脚本，但每条脚本记录均对应于某一位测试人员，因此用户与

脚本记录为一对多，将录制脚本关系抽离同上述原因。不同应用、设备可以任意组合进行脚本录制，但某条脚本记录必然对应一台设备与一款应用，因此脚本记录与应用、设备之间均为多对一的关系。设备表与应用表是为了前端更好的进行内容展示所建。

如下表 3.14所示，给出了脚本录制回放过程中涉及较多的脚本记录实体对应的script表的具体字段与含义说明。

表 3.14: script表

字段	含义	类型	属性	备注说明
script_id	脚本id	int	PK	用于对应录制脚本的id用例编号
device_udid	设备Udid	String	FK	FK，指向设备表，对应该脚本录制时使用的设备
app_id	应用id	int	FK	FK，指向应用表，对应该脚本录制时具体的应用
name	脚本名称	String	NN	脚本的名称
current_step	当前脚本步骤，即脚本步骤数	int	NN	用于脚本录制时，记录当前操作的步骤数，脚本回放时该字段可以用于判断脚本是否完成回放
dirs_location	脚本文件存放位置绝对路径	String	NN	该字段用于记录脚本每一步骤文件存放的绝对路径，为路径1,…，路径N形式存放，用于脚本回放时方便对脚本进行步骤拆分与单步回放文件提取
create_time_millis	脚本创建时间	datetime	NN	脚本记录创建的时间
status	脚本状态	int	NN,default=1	用于判断脚本记录状态,枚举类型举例：-1 脚本被删除,1 脚本可正常回放
description	脚本描述	String	NN	用于存放脚本记录相对应的测试用例，备注说明等信息

3.7　本章小结

本章主要描述了跨平台脚本录制与回放系统的整体概述，并结合应用场景做了分析。之后对系统的功能性需求与非功能性需求结合系统用例给出了具体详细的描述。然后重点介绍了脚本录制与回放系统的整体架构设计，模块组成。分别对设备管理、脚本录制、脚本回放三个主要模块的设计基于流程图进行了进一步的阐述。接着对系统中脚本回放阶段的关键点，图像匹配、布局匹配方案进行了说明。最后对数据库设计给出了数据库关键实体关系图与关键表结构设计表进行简要介绍。

第四章　跨平台测试脚本录制与回放系统详细设计与实现

4.1　设备管理模块详细设计与实现

脚本录制与回放的实现需要移动设备提前接入系统。目前跨平台测试脚本录制与回放系统支持Android平台设备进行测试脚本录制，测试脚本回放可选择Android或iOS 平台移动设备来完成。设备管理模块实现了对Android与iOS平台移动设备的统一管理，方便脚本录制与回放服务与设备之间进行交互。该模块主要从设备接入与状态更新和设备数据传输两个部分介绍具体实现细节。

4.1.1　设备接入与状态更新

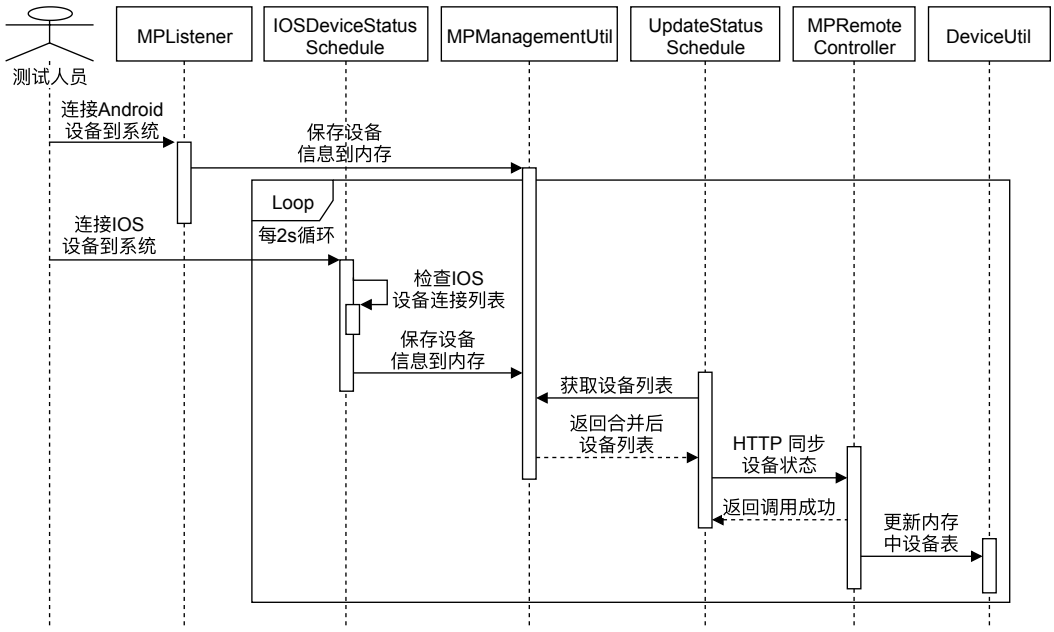

图 4.1: 设备接入与状态更新顺序图

如图 4.1所示是设备管理服务对不同平台设备接入系统及状态维护的调用顺序，由于Android与iOS 平台的底层差异，对于不同平台设备的接入与状态更新采用不同方式实现。

Android平台通过ADB工具实现服务与设备的通讯及设备状态的监控，其中监听器MPListener是对AndroidDebugBridge.IDeviceChangeListener 的实现。

新设备通过USB由测试人员手动连接至服务器，且需要设置移动设备允许当前服务器访问。监听器MPListener 获取新设备连接到当前系统事件，调用saveDeviceInfo 方法获取移动设备信息并将设备信息存储或更新到内存设备列表中。iOS平台通过libimobiledevice工具查询当前系统中连接状态设备信息。定时器IOSDeviceStatusSchedule通过该工具每2秒检查设备信息并保存或更新到内存设备列表。设备信息统一以Map 形式存储于内存，Map 结构为设备的Udid 为主键，设备类Device为值。MPManagementUtil主要功能是对设备列表Map<String,Device> 操作进行封装，对内存中设备信息操作以静态函数形式向外提供。

定时器UpdateStatusSchedule随设备管理服务同时启动，并以2秒/次的频率读取内存中当前设备列表，同时通过HTTP请求调用脚本录制回放服务中MPRemoteController提供的设备状态同步接口，发送当前设备信息列表。脚本录制回放服务中同样以Map<String,Device>结构在内存中维护设备信息列表，DeviceUtil实现对设备列表相关操作静态函数封装。脚本录制回放服务接到设备列表同步请求后会调用DeviceService及其实现类中状态更新函数UpdateStatus对DeviceUtil进行操作，完成设备状态的更新，从而保证与设备管理服务设备状态及手机状态的同步一致性。同时设备信息将存储到Mysql数据库中，以记录系统中所有连接过的设备信息。当设备掉线时，设备管理服务将该设备从设备列表中删除，同步至脚本录制回放服务中设备列表不包含掉线设备，按照新设备列表对内存中设备列表进行状态更新，将掉线设备信息从Map表中删除。保证通过前端设备管理界面能实时查看系统中连接设备，及各设备使用情况。

表 4.1: 设备接入与状态更新主要类

分类	主要包含的类
监听器类	MPListener
定时器类	UpdateStatusSchedule、IOSDeviceStatusSchedule
工具类	MPManagementUtil、DeviceUtil
控制类	MPRemoteController
服务类	DeviceServiceImpl
数据库访问类	DeviceDao
数据类	Device

设备接入与状态更新所涉及主要类如表 4.1所示，根据类别对类进行了分类，并给出不同类别下包含的类名。主要有实现监听器功能的MPListener类，实现定时器功能的UpdateStatusSchedule、IOSDeviceStatusSchedule 类，对外提供

接口服务的控制类MPRemoteController，实现具体业务功能的服务类DeviceService-Impl，对设备信息列表操作实现封装的工具类MPManagementUtil、DeviceUtil，以及数据库访问类DeviceDao和描述设备信息的设备实体类Device类。

如图 4.2所示，为主要类之间的关系，以类图的形式给出。DeviceUtil 、MPManagementUtil为两个服务器中对内存中设备列表操作进行封装的工具类，MPListener主要监听设备状态，调用MPManagementUtil完成本地内存更新，UpdateStatusSchedule为定时器，每2秒从MPManagementUtil读取当前内存中设备列表信息调用MPRemoteController向上层服务发送，保证系统中设备状态的一致性。定时器IOSDeviceStatusSchedule每2秒轮询当前系统iOS设备信息并保存。DeviceServiceImpl 为DeviceService的实现类，包含了完成当前系统设备与接收到新设备列表同步的相关服务，在更新内存设备信息的同时会调用DeviceDao向数据库中写入或更新设备信息。

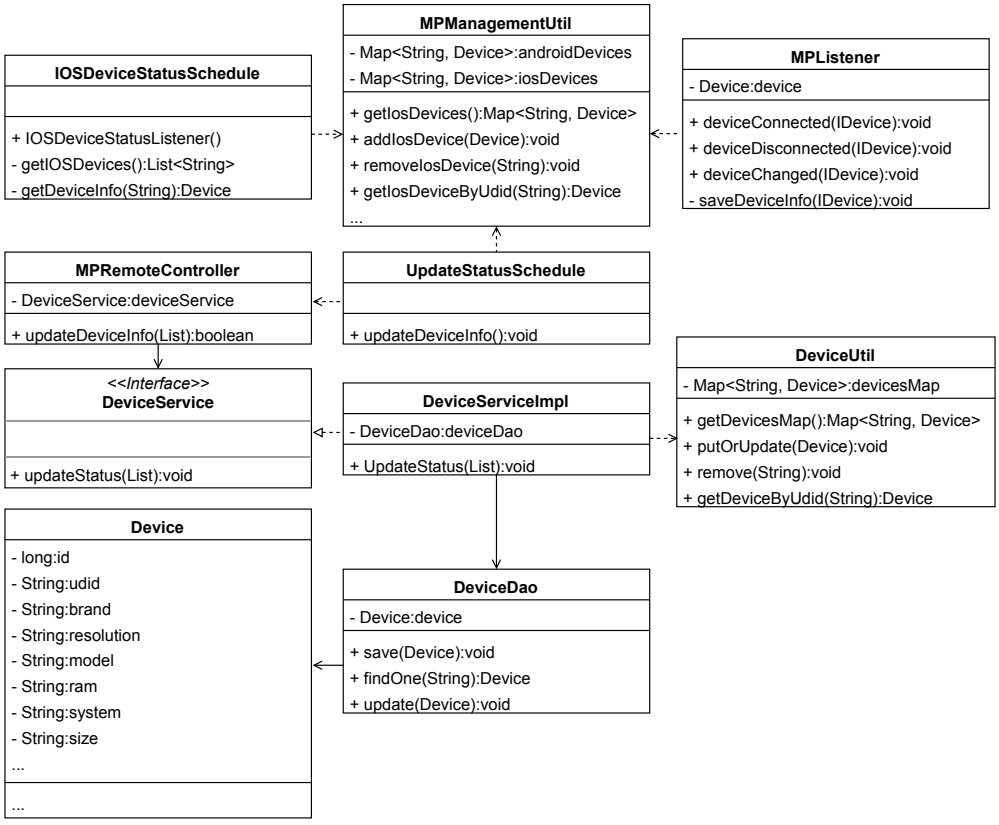

图 4.2: 设备接入与状态更新核心类图

监听器MPListener主要处理Android设备的接入与状态管理。主要实现代码如图 4.3所示，包含Android设备连接、断开及设备状态变化时的操作，同时私有函数saveDeviceInfo将当前系统设备信息存储于内存，供定时器向上同步。

```
public class MPListener implements AndroidDebugBridge.IDeviceChangeListener {
    private Device device;
    @Override
    public void deviceConnected(IDevice iDevice) {
        if(iDevice.isOnline()) {
            saveDeviceInfo(iDevice);
        }
    }

    @Override
    public void deviceDisconnected(IDevice iDevice) {
     //设备掉线 MPManagementUtil 中删除该设备}

    @Override
    public void deviceChanged(IDevice iDevice, int i) {
            //设备状态改变处理}

    private void saveDeviceInfo(IDevice iDevice) {
        device = new Device();
        device.setUdid(iDevice.getSerialNumber());
        //设备属性设置
        device.setSystem(iDevice.getProperty(DeviceInfoConstants
                                                .VERSION_RELEASE));
        device.setRam(iDevice.getProperty(MPInfoConstants.RAM));
        //将设备更新到内存中
        MPManagementUtil.addDevice(device);
    }
}
```

图 4.3: Android设备接入与设备信息存储代码

4.1.2　设备数据传输

设备数据传输作为测试人员与终端设备完成远程操控的基础，主要包括数据在Web端与设备端流动的两个过程，即界面截图从终端设备传输到Web端和测试人员Web 端操作转化为操控命令向终端设备传递。图 4.4对设备数据在系统中的传输流程进行了大致描述。

设备数据在Web端与设备间传输建立在两层Socket连接之上。测试人员在前端选择具体设备后发送HTTP请求到ScriptController，调用connectDevice方法开始建立Web端与设备之间连接。通过ScriptServiceImpl检查Netty服务端是否正常运行，如未正常运行，则会分别启动面向Web 端与设备管理服务两侧的Netty服务端。随后通过HTTP请求将设备Udid传递至设备管理服

务MiniToolService，与Udid 对应设备建立Socket连接，返回设备Banner信息，同时针对该设备启动Netty客户端，将设备Udid返回至SocketServerHandler中，调用DeviceChannelUtil将Udid与Channel键值对进行保存，完成设备与Netty客户端之间的关联。同时设备Banner 信息会逐层上传至Web端，Web端获取到设备Udid后创建Netty客户端，通过TCP 将Udid信息发送至WSSocketServerHandler，同时DeviceChannelUtil完成对Udid与Web端Channel键值对的记录，Web端到设备的两层Socket连接借助Netty框架搭建完成。

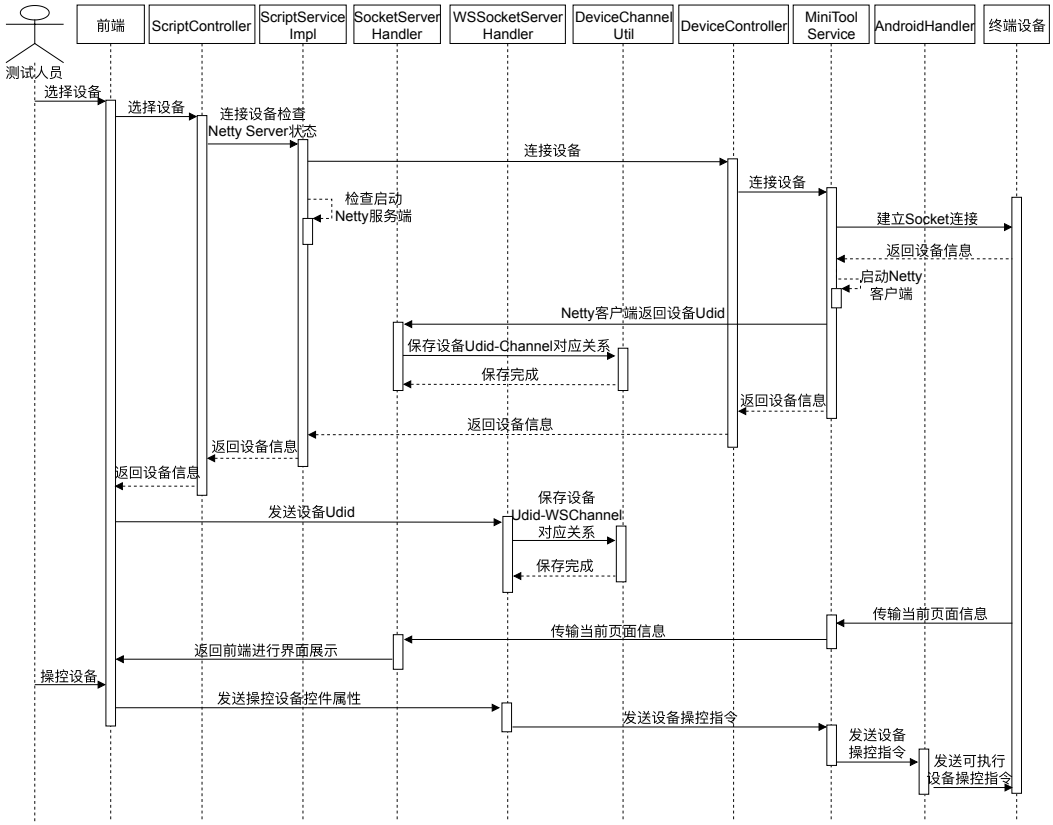

图 4.4: 设备数据传输顺序图

脚本录制回放服务作为中转平台，完成了Web端与设备的信息交换。首先是界面截图数据从终端设备到Web端传输。设备管理服务使用TCP协议将获取到的当前设备界面截图通过Channel不断向脚本录制回放服务传输，该服务通过Channel获取对应设备Udid，进而找到对应WSChannel将截图向Web端转发，实现设备在Web端界面同步展示。然后Web端操作信息向设备传输过程则是操作信息及Udid通过WebSocket协议发送至脚本录制回放服务，将操作转化为设备可执行指令，随后获取与Udid 对应的Channel实现操作指令到设备管理服务的传递，最终完成设备界面与测试人员操作的连动。

表 4.2: 设备数据传输主要类

分类	主要包含的类
处理器类	SocketServerHandler、WSSocketServerHandler、AndroidHandler
工具类	DeviceChannelUtil、ExecuteUtil
控制类	ScriptController、DeviceController
服务类	ScriptServiceImpl、MiniToolService

如表 4.2 所示，给出了设备数据传输过程中主要涉及的类的信息。主要包括实现Netty框架数据接收与处理功能的处理器类SocketServerHandler、WSSocketServerHandler、AndroidHandler。记录设备Udid与Channel对应关系的工具类DeviceChannelUtil，iOS平台操作封装工具类ExecuteUtil。接口封装提供Web端调用设备的控制类ScriptController，对调用设备执行操作进行了接口封装，提供脚本录制回放服务调用的控制类DeviceController。具体业务逻辑服务类ScriptServiceImpl、MiniToolService。

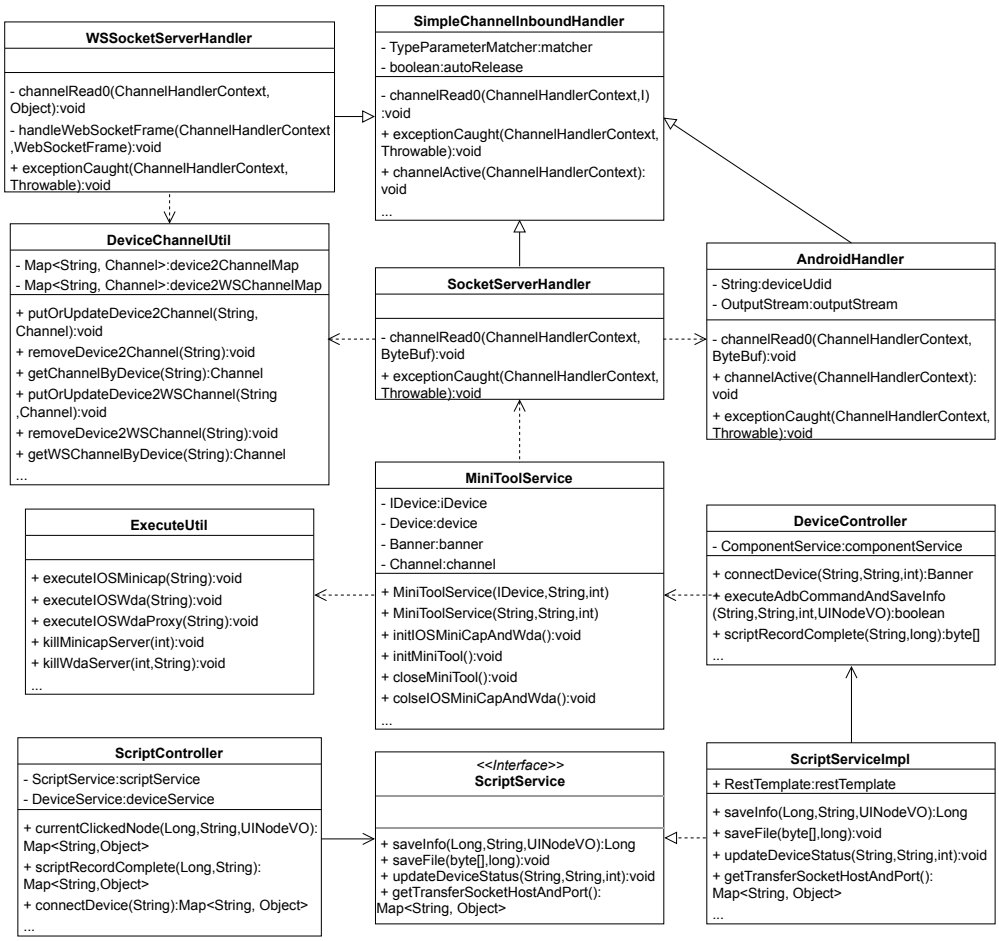

图 4.5: 设备数据传输核心类图

图 4.5给出了设备数据传输过程中涉及类的关系。DeviceChannelUtil记录了设备管理服务、Web端两侧Netty客户端与设备关系。ExecuteUtil针对iOS平台封装了iOS-MiniCap及WDA工具初始化与销毁过程。ScoketServerHandler、WSScoketServerHandler及AndroidHandler继承自SimpleChannelInboundHandler类，主要实现数据接收、处理及定向转发，实现了Web端到设备的双层Netty闭环通讯。ScriptServiceImpl为ScriptService实现类，在处理设备数据到设备管理服务器的同时检查Netty服务端，并在异常情况下重启服务，保证Netty服务端的正常运行。MiniToolService封装了在建立设备连接时获取设备Banner信息的过程，在得到设备详细信息后启动设备端Netty客户端程序开始构建数据交互闭环。

```
public void startSocket() {
        EventLoopGroup bossGroup = new NioEventLoopGroup();
        EventLoopGroup workerGroup = new NioEventLoopGroup();
        try {
            ServerBootstrap serverBootstrap = new ServerBootstrap();
            serverBootstrap.group(bossGroup, workerGroup)
                .channel(NioServerSocketChannel.class)
                .option(ChannelOption.SO_BACKLOG, 1000)
                .childOption(ChannelOption.SO_KEEPALIVE, true)
                .childHandler(new ChannelInitializer<SocketChannel>() {
                 @Override
                 protected void initChannel(SocketChannel socketChannel) throws
 Exception {
                            //初始化 Netty 服务端
                            ChannelPipeline ch = socketChannel.pipeline();
                            ch.addLast(new LengthFieldBasedFrameDecoder(Integer.
MAX_VALUE, 0, 4, 0, 4));
                            ch.addLast(new LengthFieldPrepender(4));
                            ch.addLast(new SocketServerHandler());
                            // 注册 SocketServer 面向设备服务器
                            // WSSocketServer 将 SocketServerHandler
                            // 替换为WSSocketServerHandler，面向前端
                    }
            });
            ChannelFuture channelFuture = serverBootstrap.bind(1800).sync();
            isSocketStart = true;
            channelFuture.channel().closeFuture().sync();
        } catch (InterruptedException e) {
            //中断异常处理，将当前线程中断
        } finally {
            //资源回收。优雅回收 bossGroup、workerGroup 资源
        }}
```

图 4.6: Netty服务端初始化代码

Netty框架的使用在设备数据传输的过程中具有至关重要的意义。且系统设计了两层Netty连接，将对终端设备的管理、真机交互与脚本的信息处理等业务流程进行了分割处理，提高了整个系统的可扩展性。系统中反复涉及了Netty客户端与服务端的创建及设备与客户端的对应匹配，对以面向设备管理服务器的Netty服务端初始化过程给出具体代码实现如图 4.6所示。面向前端的初始化过程大致相同，将处理器进行置换即可。

4.2　脚本录制模块详细设计与实现

4.2.1　脚本单步操作记录

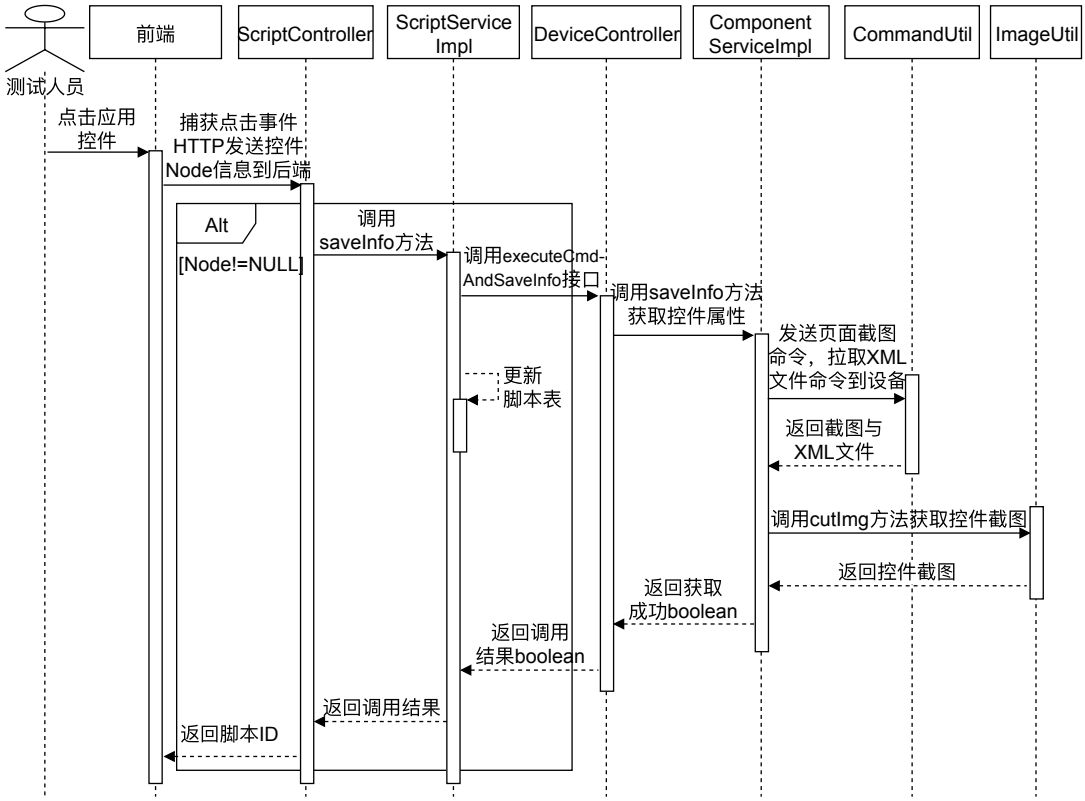

图 4.7: 脚本单步操作记录过程顺序图

脚本的录制是在Android平台下进行的，如图 4.7所示，给出了脚本单步操作记录流程的顺序图。

脚本录制过程是在Web端建立与设备连接之后，由测试人员点击开始录制按钮触发。测试人员在前端点击应用控件进行脚本录制，前端通过监听鼠标Mouseup事件在当前页面中查找最适合的Node节点，将获取的Node信息及Script_id编号同时发送至后端ScriptController接口。脚本录制过程首次

操作无Script_id，因此默认首次操作Script_id值为-1。脚本录制回放服务接收到Node及Script_id，若Node不为空，ScriptServiceImpl中saveInfo方法根据Script_id值进行创建或更新数据库Script表中记录，同时调用DeviceController接口将Node信息、设备Udid及当前执行步骤数等信息向设备管理服务传递。

设备管理服务获取到控件信息后进行控件属性的提取操作，通过ComponentServiceImpl中saveInfo方法获取包括当前页面截图、XML文件、控件截图等信息，并返回给脚本录制回放服务。CommandUtil对设备操作进行封装，以设备Udid及Android设备可执行指令为参数，通过执行Shell命令获取设备执行结果并返回。saveInfo方法通过调用CommandUtil执行界面截图与XML 文件获取操作，界面截图及界面XML文件即存在于对应设备内存，随后通过调用IDevice.pullFile方法将设备中文件拉取到本地服务器文件系统，单步操作XML文件及界面截图获取完成。接着调用ImgUtil封装了获取控件截图的cutImg 方法完成对控件截图的获取。该方法以当前页面截图及Node中控件位置信息为参数，使用ImageIO对图像进行处理，截取指定区域的图像并存储。随后逐层对结果进行返回，直至ScriptController 接口向Web端会返回当前Script_id。之后所有Web端操作请求携带该Script_id 信息，以区分不同脚本录制过程。

表 4.3: 脚本单步操作记录主要类

分类	主要包含的类
控制类	ScriptController、DeviceController
服务类	ScriptServiceImpl、ComponentServiceImpl
工具类	CommandUtil、ImageUtil
数据库访问类	ScriptDao
数据类	Script

表 4.3 给出了脚本单步操作过程中所涉及的主要的类的类型及类名称。包括面向前端提供脚本操作信息记录接口的控制类ScriptController，设备管理服务用于接收操作指令接口的控制类DeviceController。服务类处理不同业务逻辑，ScriptServiceImpl 负责脚本记录生成、更新至数据库及向设备管理服务发送当前操作信息并对返回脚本截图及XML等属性进行存储，ComponentServiceImpl负责将获取到的控件信息转为设备可执行指令并发送到设备，处理截图及XML文件等从设备到服务器的获取过程并对结果进行返回。工具类CommandUtil实现对设备操作的封装，通过Shell 命令完成与设备的交互。ImageUtil则对控件截图提取函数进行封装。数据库访问类ScriptDao提供服务对数据库Script 表的操作函数。数据类Script为测试脚本的实体类。

脚本单步操作记录核心类图如图 4.8所示。ScriptController面向Web端提供当前操作记录接口，其中currentClickNode方法用于记录当前操作控件对象属性，调用ScriptService接口实现对脚本单步操作信息的记录，具体业务由ScriptServiceImpl 实现。ScriptServiceImpl调用ScriptDao 完成对脚本记录的数据库更新，通过RestTemplate发起Http请求调用DeviceController从移动设备处获取操作记录所需属性。

ComponentServiceImpl是对ComponentService抽象类的实现，saveInfo方法负责与设备交互，获取操作记录所需截图及XML信息。通过调用CommandUtil实现对设备界面截图及XML文件的获取，调用ImageUtil提供cutImg方法以控件Node信息及界面截图为参数获取当前操作控件截图并保存，完成对脚本的单步操作记录。

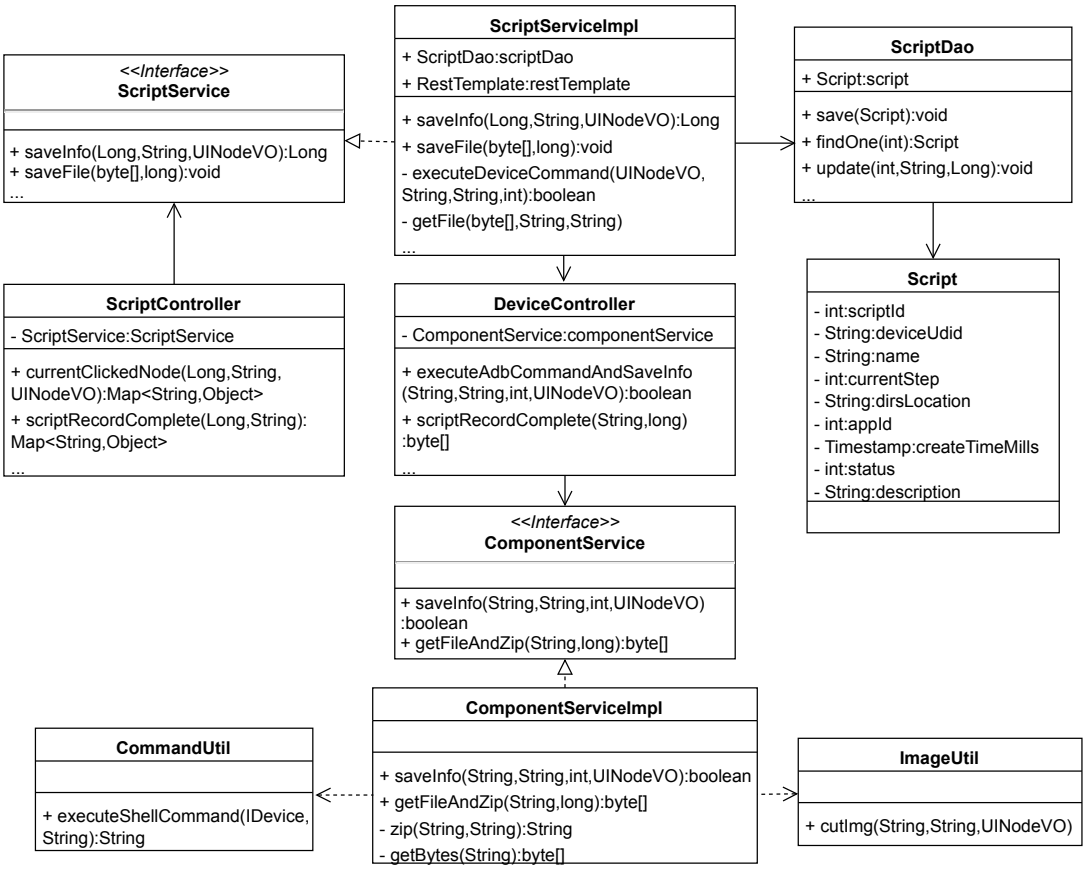

图 4.8: 脚本单步操作记录核心类图

对于控件的截图由于不能直接从设备中截取，因此需要将设备当前界面截图与控件边界坐标信息结合，通过对图片进行区域截取获取到控件截图文件，该过程是通过ImageIO库对图片进行截取操作，具体代码如图 4.9所示。

```
public static void cutImg(String inputImagePath, String outputImagePath, UINodeVO
uiNode) {
    …// 初始化 FileInputStream，ImageInputStream 变量代码省略
    try {
        fis = new FileInputStream(inputImagePath);
        // 返回能够解析 png 格式图片的 ImageReader 的 Iterator
        Iterator<ImageReader> it=ImageIO.getImageReadersByFormatName("png");
        ImageReader imageReader = it.next();
        // 获取图片流 iis:读取源.true:只向前搜索.将它标记为“只向前搜索”。
        iis = ImageIO.createImageInputStream(fis);
        imageReader.setInput(iis, true);
        // 描述如何对流进行解码的类
        ImageReadParam param = imageReader.getDefaultReadParam();
        Rectangle rectBound = new Rectangle(uiNode.getxPosition(),uiNode
                              .getyPosition(), uiNode.getWidth(), uiNode.getHeight());
        // 提供一个 BufferedImage，将其用作解码像素数据的目标。
        param.setSourceRegion(rectBound);
        // 它作为一个完整的 BufferedImage 返回。
        BufferedImage bufferedImage = imageReader.read(0, param);
        ImageIO.write(bufferedImage, "png", new File(outputImagePath));
    }…//异常处理及对 fis iis 资源进行回收
}
```

图 4.9: 获取控件截图方法代码

4.2.2 脚本生成

脚本的生成过程如图 4.10所示，测试人员在完成脚本录制后在Web端点击完成录制按钮触发该流程。Web端采用HTTP调用将Script_id及设备Udid以参数形式传递于ScriptController，之后通过RestTemplate调用DeviceController接口将信息传至设备管理服务完成脚本录制并获取录制完成的脚本文件。

设备管理服务DeviceController接口获取到脚本ID及设备Udid信息后调用ComponentService接口，将与Script_id 对应的录制过程脚本文件进行Zip 压缩并按照“Udid_ScriptID”形式命名根目录。压缩完成后获取到Zip格式压缩文件并以字节流形式逐层向上返回直至ScriptController，接着通过对ScriptServiceImpl调用，将获取到的字节流文件向Zip格式压缩文件进行转化，转化完成后对文件解压获取脚本文件。脚本文件在根目录下不同操作单独目录存储的层次结构存放在脚本录制回放服务器，并在文件解压后获取脚本文件单步操作文件路径通过ScriptDao更新至数据库，完成脚本录制及脚本文件的生成。脚本Zip格式文件路径作为调用结果返回至Web端，测试人员可以在Web端点击下载脚本文件至本地进行录制脚本的查看。

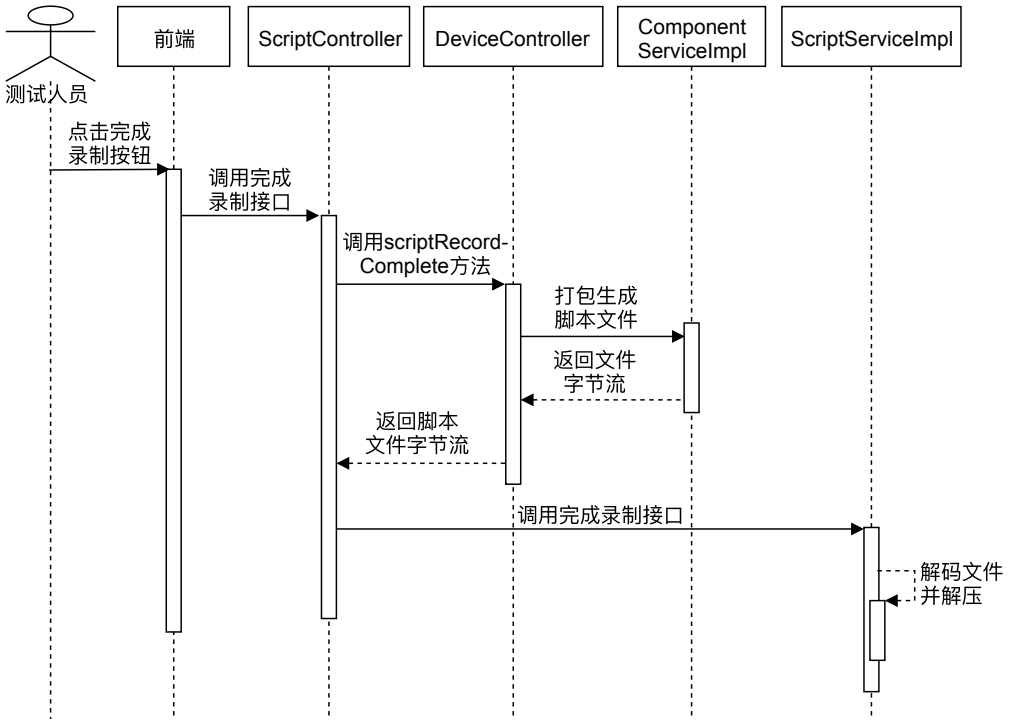

图 4.10: 脚本生成顺序图

表 4.4给出了脚本生成过程中涉及到的主要类。实体类Script及数据库访问类ScriptDao 实现对数据库的访问及操作。控制类ScriptController提供脚本生成接口供Web 端完成脚本录制操作时调用，DeviceController提供单步操作文件集合Zip压缩处理供脚本录制回放服务调用。服务类ScriptServiceImpl、ComponentServiceImpl 主要实现脚本生成相关业务逻辑。

表 4.4: 脚本生成主要类

分类	主要包含的类
控制类	ScriptController、DeviceController
服务类	ScriptServiceImpl、ComponentServiceImpl
数据库访问类	ScriptDao
数据类	Script

核心类之间关系由图 4.11给出。ScriptController提供scriptRecordComplete方法调用ScriptService完成脚本的录制及生成。实现类ScriptServiceImpl中saveFile方法完成脚本生成具体逻辑，将获取的脚本字节流文件转为Zip文件并解压，同时更新文件路径至数据库。DeviceController调用ComponentService压缩生成脚本文件Zip包并转为字节流返回，实现类ComponentServiceImpl中getFileAndZip 方法实现具体压缩及字节流转化业务逻辑。

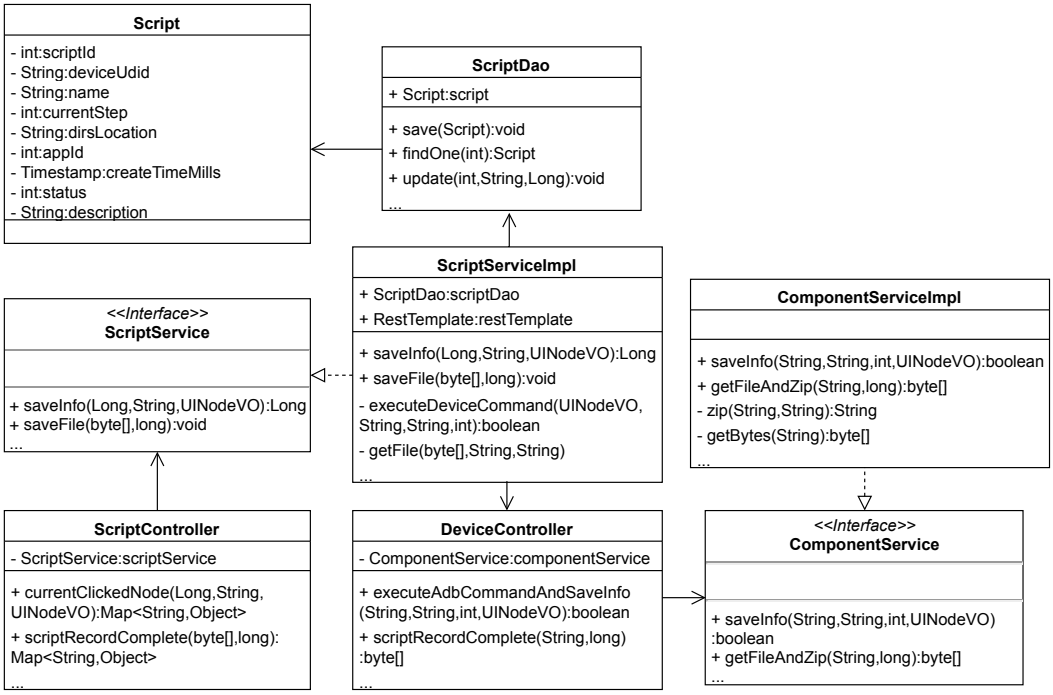

图 4.11: 脚本生成核心类图

在脚本生成过程中比较关键部分为脚本文件的压缩传输，其中压缩采用Zip4j工具实现对多层脚本文件的压缩处理。zip方法具体代码如图 4.12 所示。

```
private String zip(String src, String dest) {
        File srcFile = new File(src);
        if(!srcFile.exists()) {srcFile.mkdirs();}
        ZipParameters parameters = new ZipParameters();
        // 设置压缩方式、压缩级别
        parameters.setCompressionMethod(Zip4jConstants.COMP_DEFLATE);
        parameters.setCompressionLevel(Zip4jConstants.DEFLATE_LEVEL_NORMAL);
        try {
            ZipFile zipFile = new ZipFile(dest);
            if (srcFile.isDirectory()) {
                // 若无外层文件夹，即直接把给定目录下的文件进行压缩,无目录结构
                zipFile.addFolder(srcFile, parameters);
            } else {zipFile.addFile(srcFile, parameters);}
            return dest;
        } // 处理异常
        return null;
    }
```

图 4.12: Zip方法代码

4.3　脚本回放模块详细设计与实现

4.3.1　脚本单步操作回放

录制过程所生成的脚本文件是对测试人员单步操作过程的一个记录，因此在回放过程中仍然分步对脚本进行回放，循环脚本单步回放过程即可完成对脚本文件的回放。脚本的单步操作回放对于脚本回放过程至关重要。脚本的单步回放过程如图 4.13所示。

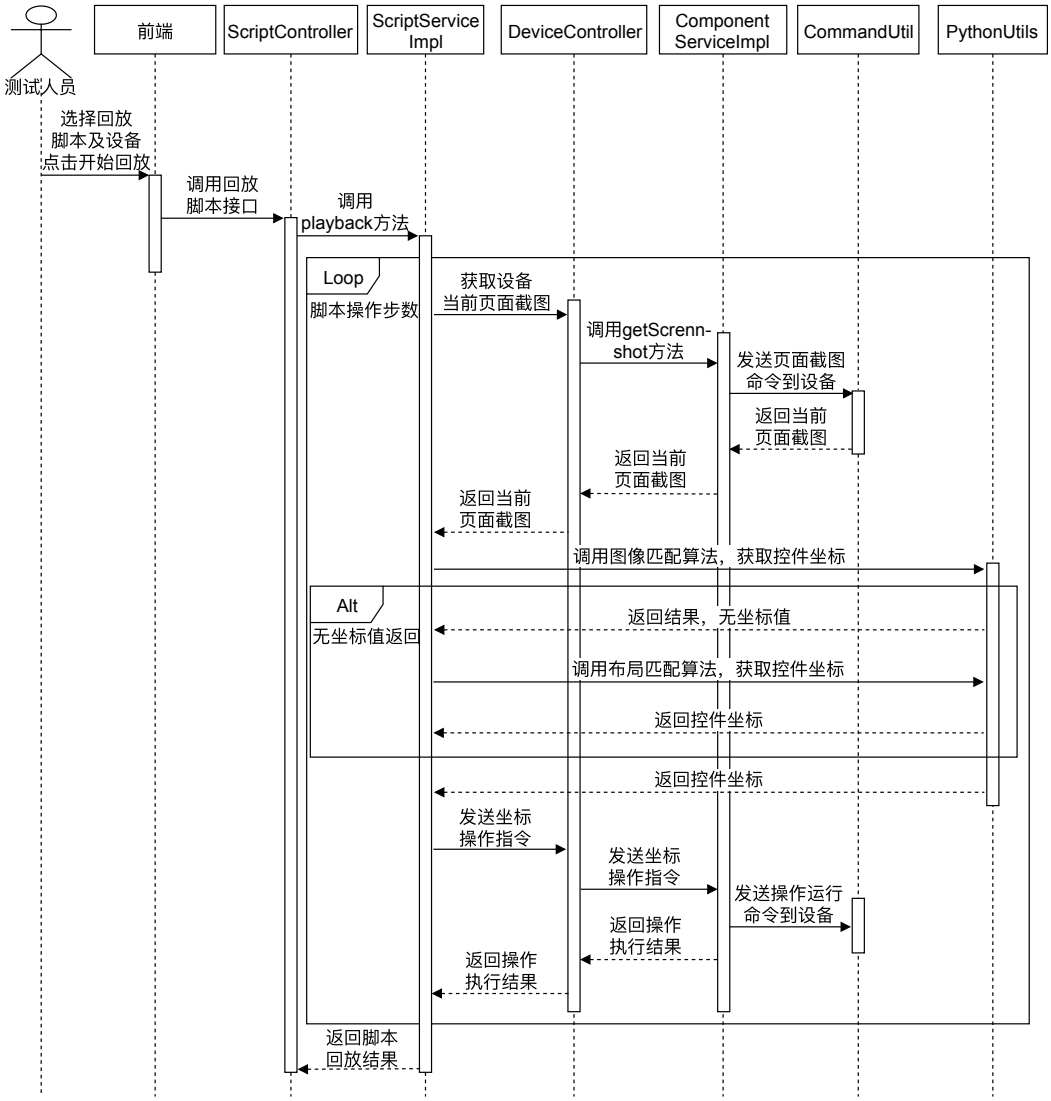

图 4.13: 脚本单步操作回放顺序图

测试人员选择脚本及设备建立连接后点击开始回放即前端向后端ScriptController 发送脚本回放请求，ScriptController将获取到的Script_id与设备的Udid一同传入业务逻辑处理部分ScriptServiceImpl，该服务类即从数据库中获取对应

脚本记录的操作步数及每步脚本文件对应存放位置的绝对路径，之后对脚本操作开始循环进行单步回放。单步回放操作首先会调用DeviceController接口获取当前设备页面截图，界面截图的获取是通过ComponentServiceImpl服务类中的getScreenshot方法实现，该方法通过CommandUtil中封装的获取设备命令的函数与设备进行交互完成当前界面截图的获取，随后逐层返回到ScriptServiceImpl中。由于图像匹配、布局匹配均是由Python编写，因此两种方式程序的调用通过PythonUtils工具类进行了调用封装。在获取到控件截图与当前页面截图后通过对PythonUtils的调用进行图像匹配与布局匹配定位，得到控件坐标的返回值，两种定位方法具体实现在下面小节给出。获取到控件在新设备页面的位置坐标后通过DeviceController、ComponentServiceImpl 等调用完成到设备上的命令执行，完成脚本单步操作回放的执行。

脚本单步操作回放过程中涉及到的主要类的信息如表 4.5 所示。包括面向Web端提供脚本回放接口的ScriptController 控制类，面向脚本录制回放服务提供获取当前设备界面截图与分发、执行指令接口的DeviceController控制类。ScriptServiceImpl、ComponentServiceImpl 是服务类，前者是对脚本回放业务流程的具体实现，后者为面向设备截图、指令分发执行的具体业务实现。工具类PythonUtil主要对系统调用图像匹配、布局匹配Python 函数流程进行了封装，CommandUtil是对设备执行指令操作进行封装。数据库访问类ScriptDao提供了对数据库Script表的访问操作支持，数据类Script是测试脚本实体类，将测试脚本相关特征属性进行提取。

表 4.5: 脚本单步操作回放主要类

分类	主要包含的类
控制类	ScriptController、DeviceController
服务类	ScriptServiceImpl、ComponentServiceImpl
工具类	CommandUtil、PythonUtil
数据库访问类	ScriptDao
数据类	Script

图 4.14 给出脚本单步操作回放过程核心类之间的关系。ScriptController通过scriptRecordPlayback方法接收Web端对具体测试脚本的回放HTTP请求，通过调用ScriptService接口中的playback方法对脚本记录操作依次循环执行脚本单步操作回放流程，ScriptServiceImpl类是对脚本回放接口的具体业务逻辑实现，在单步脚本操作回放过程中调用对PythonUtil 中invok方法分别实现图像匹配与布局匹配两种定位方式获取控件坐标位置，并通过executeDeviceCommand方法发送至设备管理服务实现操作在具体设备上的回放执行。单步操作回放时调

用ScriptDao根据script_id在数据库中查找相应脚本记录，获取脚本操作步骤及相关文件存储路径，查找结果以Script实体形式返回。

DeviceController是设备管理服务在脚本回放流程中对上层提供获取当前设备界面截图及执行指令的接口。由于脚本回放需要获取回放设备当前界面进行控件的定位，getCurrentScreenshot方法即通过对ComponentService调用实现设备截图获取。ComponentServiceImpl是对具体业务逻辑的实现，对Android与iOS不同平台分别采用ADB与WDA工具获取设备当前界面截图并返回。CommandUtil工具类是辅助完成对Android平台设备界面截图获取。

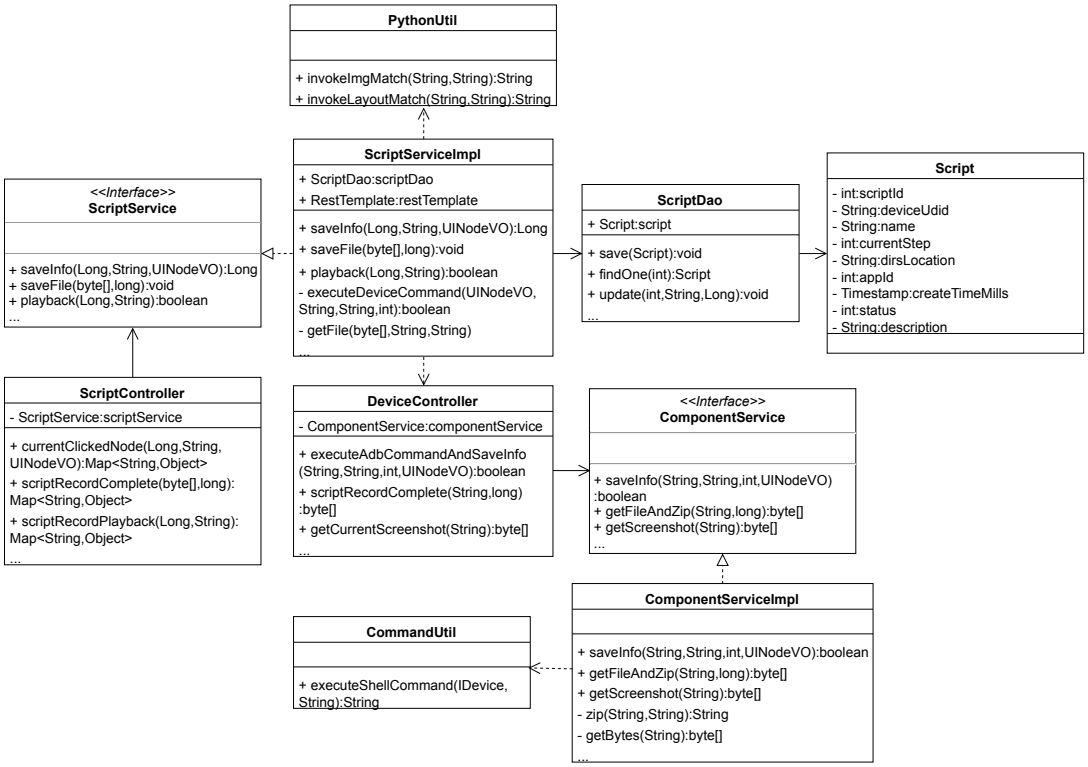

图 4.14: 脚本单步操作回放核心类图

脚本回放是对脚本单步操作记录的循环回放，Web端通过HTTP调用将脚本ID与回放设备Udid以参数形式传递至后端，根据脚本ID获取数据库中脚本记录信息，对脚本中单步操作记录进行控件定位及设备操作回放过程。优先通过系统调用图像匹配程序进行控件定位，异常情况时则通过布局匹配d 定位控件，获取匹配控件坐标后发送至设备实现真机回放。控件定位需要获取当前设备界面截图，该过程通过设备管理服务getCurrentScreen方法调用实现。当两种方式均未能定位控件，即循环中断返回脚本回放异常结果至Web端，展示脚本回放异常。具体脚本回放实现代码如图 4.15 所示。

```
public boolean playback(Long scriptId, String deviceUdid) {
        …//获取每步操作的文件存储路径集合代码省略
        Pattern locationPattern = Pattern.compile("\\[-?(\\d+),-?(\\d+)\\]");
        for(int i = 0;i < dirsLocation.size();i++) {
            byte[] bytes = getCurrentScreen(deviceUdid);
            if(bytes != null) {
                getFile(bytes, dirsLocation.get(i) + "/", "curFullscreen.png");
                //读取当前步骤控件截图位置及当前设备页面截图位置
                String elePath = dirsLocation.get(i) + "/element.png";
                String curscPath = dirsLocation.get(i)+"/curFullscreen.png";
                //调用 python 程序执行图像匹配
                String result = PythonUtils.invokeImgMatch(elePath,curscPath);
                Matcher m = locationPattern.matcher(result);
                if(m.matches()) {
                    String command = "input tap "+m.group(1)+" "+m.group(2);
                    executeTapCommand(deviceUdid, command);
                } else { //调用 python 程序执行布局匹配
                 String result = PythonUtils.invokeLayoutMatch(elePath, curscPath);
                 if(m.matches()) {
                    …// 同上发送控件坐标位置到设备，执行回放操作
                 } else {//未匹配到控件则跳出循环，返回执行失败}
        }}}
        …//返回执行结果代码省略
    }
```

图 4.15: 脚本回放业务逻辑代码

4.3.2　图像匹配定位

图像匹配定位算法采用Python调用OpenCV 图像处理库实现，主要过程包括了特征提取、特征点匹配、消除错配、计算畸变，最后将匹配到的图像区域转化为区域中心坐标点输出。具体处理流程如图 4.16所示，算法中涉及的具体设计已在第三章实现方案中说明，此处不再赘述。以下对各部分实现代码进行具体说明。

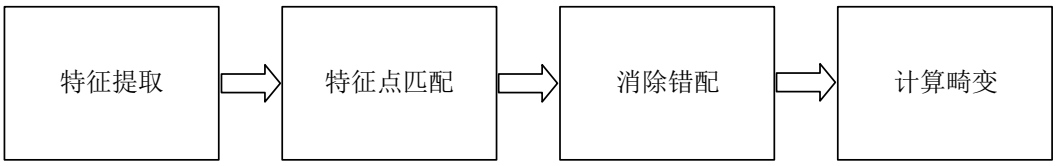

图 4.16: 图像匹配定位处理流程图

特征提取与特征点匹配。以控件图像与界面图像为输入，将图像通过cvtColor颜色空间转换函数变为灰度图优化图像质量作为后续处理的对象。随

后创建SIFT算子，detectAndCompute函数分别检测、计算出图像的特征点及描述子集合，完成对图像特征的提取。FLANN特征匹配算法采用KDTREE索引，为了保证算法效率与匹配效果的平衡性，设置FLANN算法trees树的数量为5，checks递归遍历次数为50。通过KNN匹配控件图像特征点集中的每个特征点在界面图像特征点集中最近的两个点，完成特征点匹配预选。图 4.17给出了具体处理过程的代码实现。

```
# 将图像转换为灰度图
imgElementGray = cv2.cvtColor(imgElement, cv2.COLOR_BGR2GRAY)
imgFullScreenGray = cv2.cvtColor(imgFullScreen, cv2.COLOR_BGR2GRAY)
# 创建 sift 引用，并计算各自特征点，描述集
sift = cv2.xfeatures2d.SIFT_create()
imgEle_keyPoint, imgEle_des = sift.detectAndCompute(imgElementGray, None)
imgFul_keyPoint, imgFul_des = sift.detectAndCompute(imgFullScreenGray, None)
# Flann 特征匹配
FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(imgEle_des, imgFul_des, k=2)
```

图 4.17: 特征提取与特征匹配主要代码

由于上述过程匹配到的特征点存在较高的误匹配情况，因此通过图4.18中给出的比率测试与RANSAC算法进行匹配点的进一步筛选，并计算两个特征点集中确定的图像区域之间的单应矩阵，即计算畸变，这一过程是由findHomography方法实现。

```
#比率测试筛选除去错配
goodMatch = []
for m, n in matches:
    if m.distance < 0.50 * n.distance:
        goodMatch.append(m)
#计算畸变，得出单应矩阵，计算匹配区域位置坐标
srcPts=np.float32([imgEle_KP[m.queryIdx].pt for m in goodMatch]).reshape(-1, 1, 2)
dstPts=np.float32([imgFul_KP[m.trainIdx].pt for m in goodMatch]).reshape(-1, 1, 2)
M, mask = cv2.findHomography(srcPts, dstPts, cv2.RANSAC, 5.0)
matchesMask = mask.ravel().tolist()
h, w = imgElementGray.shape
pts = np.float32([[0, 0], [0, h - 1], [w - 1, h - 1], [w - 1, 0]]).reshape(-1, 1, 2)
dst = cv2.perspectiveTransform(pts, M)
```

图 4.18: 消除错配与计算畸变主要代码

4.3.3　布局匹配定位

布局匹配定位基于REMAUI工具优化改进实现，利用图像分割Canny算法、OCR技术对不同设备上同一应用的相同界面进行布局分割处理，将工具处理得到的页面布局轮廓进行筛选、编号，通过记录录制界面控件布局编号，在新设备页面布局查找相同编号轮廓，定位控件，将该轮廓中心坐标返回，前后页面分割结果轮廓的匹配性则是该方案准确率的关键因素，因此对于结果筛选的处理则变得尤为关键。图 4.19给出了布局匹配的业务逻辑流程代码。

```
# 利用 canny 算法对截图进行处理
canny = Canny()
# canny 处理得到轮廓边点集
dst_edge = canny.findEdge(img_gray)
# 膨胀，合并临近点
dst_edge_dilate = canny.addDilate(dst_edge)
contourAnalysis = ContourAnalysis()
# findContours 查找外轮廓，参数设置 RETR_TREE，CHAIN_APPROX_SIMPLE
contours = contourAnalysis.findContoursWithCanny(dst_edge_dilate)
# 对轮廓进行筛选
contoursOutput = contourAnalysis.analyze(dst_edge_dilate, contours)
# 处理布局结构
hierarchyProcessor = ViewHierarchyProcessor(contoursOutput.rootView, img_color,
canny)
hierarchyInfo = hierarchyProcessor.process()
# 通过 tesseract 处理文本信息，导出 ocr 布局
tesseractOCR = TesseractOCR(dst_denoised, dipCalculator, "English")
textProcessor = TextProcessor(img_color, dst_denoised, hierarchyInfo.
biMapViewRect, tesseractOCR, dipCalculator)
# 去除无效的文本区域
textInfo = textProcessor.processText(CColor.Red)
# 添加 ocr 获取的布局信息到 canny 处理结果中，结果合并
hierarchyProcessor.addTextToHierarchy(textInfo)
```

图 4.19: 布局匹配业务逻辑流程主要代码

findContours是对canny分割出的图像轮廓点集进行处理，通过参数设置为RETR_TREE即检查点集返回检测到的所有的轮廓，并对所有轮廓建立一个等级树结构，存入hierarchy向量。另外参数CHAIN_APPROX_SIMPLE表示对提取的轮廓仅保留拐点信息，把所有的轮廓拐点信息存入contours向量。函数处理结果返回两个向量，且两个向量元素一一对应，hierarchy向量内每个元素均有4个int变量，代表本轮廓的后一个轮廓、前一个轮廓，外轮廓、内轮廓在contours向量中的索引编号，在某项缺失时该项值为-1。通过该信息，

在analyze方法中，添加了对轮廓的筛选，保证轮廓的有效性及尽量减少轮廓数量，增加处理不同设备相同应用图片时布局刻画的一致性。其中对工具改进、优化，轮廓筛选的具体代码如图 4.20所示。

```
def pretreatment(self, width, height, contours, hierarchy):
    …// 获取 Canny 分割结果，初始化变量代码省略
    for index, component in enumerate(canny_result):
        currentContour = component[0]
        currentHierarchy = component[1]
        x, y, w, h = cv2.boundingRect(currentContour)
        # 对长或宽小于阈值的轮廓进行筛除，阈值此处设为 width*0.02
        isAccept = self.filterToShortEdgeBox(w, h, width*0.02, width*0.02)
        parentIsOutsideBox = False #是否为外边框轮廓标志位
        if currentHierarchy[3] > 0:
            oX, oY, oW, oH=cv2.boundingRect(canny_result[currentHierarchy[3]][0])
            if oW >= 0.9 * width: # 对宽小于 90%界面宽度的轮廓的内轮廓筛除
                parentIsOutsideBox = True
        else:
            parentIsOutsideBox = True
        if isAccept:
            if w < 0.9 * width and not parentIsOutsideBox:
                isAccept = False
        if not isAccept:
            temp.append(index)
    …// 过滤掉 temp 中筛选的不合条件轮廓，对轮廓结果进行返回代码省略
```

图 4.20: 轮廓筛选主要代码

filterToShortEdgeBox为自定义函数，主要功能为筛选出轮廓边长长度远小于当前屏幕宽度的轮廓。一般应用内该种形状轮廓不被设计为功能性控件，因此可以对该种轮廓进行过滤。同时对非外边框轮廓内部的轮廓进行筛除。最终得到的结果与OCR 结果结合，对轮廓编号匹配。

4.4　系统测试与运行效果分析

测试脚本录制与回放系统基于以上章节中各个模块设计与实现，本小节主要通过对实现的系统从单元测试与功能测试两方面进行描述。单元测试主要针对设备管理模块、脚本录制模块与脚本回放模块中类所包含的函数进行测试，保证代码的质量。功能测试通过对系统中实现的功能设计对应测试用例进行测试，检查系统运行是否满足业务需求。通过选取不同移动应用及移动设备，设计脚本录制与回放实验对系统运行效果进行分析，验证本系统的实际可用性。

4.4.1　系统测试环境

测试之前需要对系统进行部署，为模拟用户真实使用场景，系统部署采用与线上同规格设备及软件进行测试环境搭建。系统采用多服务器对不同服务进行分布式部署，图像识别服务基于Python实现，设备管理服务、脚本录制与回放服务均基于Java实现，其中由于设备管理服务需要对iOS系统进行支持，因此选用macOS系统进行部署，其它服务均采用Ubuntu Server服务器进行部署。系统采用MySQL数据库，通过Nginx实现反向代理与负载均衡。系统支持Android与iOS系统设备，通过USB 与设备管理服务器连接实现通讯。系统具体测试环境配置信息如表 4.6所示。

表 4.6: 系统测试环境

环境项	运行环境
操作系统	macOS Mojave(10.14.3)、Ubuntu 16.04
CPU	Intel Core i7 3.2GHz
内存	16G
JDK版本	jdk1.8.10_131
数据库版本	MySQL 5.7.18
Python版本	Python 3.7.2
移动设备	华为、三星等主流Android机型，iPhone5s-iPhone8部分机型
其他软件	Nginx、Tesseract 4.0.0、Xcode 10.1、Android SDK

4.4.2　单元测试

单元测试是指开发者通过编写测试代码对程序中最小功能单元进行检测与验证。为保证服务端代码质量，本系统通过单元测试对各模块接口进行测试验证。由于系统服务端采用Java进行开发，因此选用Junit 作为单元测试框架，同时采用Jenkins对项目进行持续集成，完成系统项目的构建、部署及测试等过程，展示系统单元测试结果。

对于系统的单元测试主要包括设备管理模块、脚本录制模块与脚本回放模块，分别对应于设备管理与脚本录制与回放两个不同的项目。通过公司内部Jenkins持续集成平台，针对以上两个项目分别创建不同Job 添加源码配置及构建触发器配置实现项目的一键构建部署。为方便Junit单元测试结果报告的生成与展示，需要在Job的触发器配置项中选择构建后操作Publish test result report，并添加报告生成路径。

设备管理项目单元测试运行结果如图4.21所示。主要针对项目中Controller、Service、Common、Configure 和Utils包中涉及的函数方法进行测试。测试报告

显示设备管理项目共进行了107个单元测试，通过率为100%，无失败或跳过的单元测试。

Test Result

0次失败 (±0)

107个测试 (±0)

花了

添加说明

所有的测试

Package	花的时间	失败（区别）	跳过（区别）	Pass（区别）	总数（区别）
com.mooctest.mobilemanagement.controller	1.42 秒	0	0	15	15
com.mooctest.mobilemanagement.service	3.51 秒	0	0	34	34
com.mooctest.mobilemanagement.common	1.83 秒	0	0	24	24
com.mooctest.mobilemanagement.configure	0.78 秒	0	0	8	8
com.mooctest.mobilemanagement.utils	2.37 秒	0	0	26	26

图 4.21: 设备管理项目单元测试报告

脚本录制与回放项目单元测试运行结果如图 4.22所示。主要针对项目中Controller、Service、Common 和Utils包中涉及的函数方法进行测试。测试报告显示设备管理项目共进行了102个单元测试，通过率为100%，无失败或跳过的单元测试。

Test Result

0次失败 (±0)

102个测试 (±0)

花了

添加说明

所有的测试

Package	花的时间	失败（区别）	跳过（区别）	Pass（区别）	总数（区别）
com.mooctest.scriptplatform.controller	2.35 秒	0	0	25	25
com.mooctest.scriptplatform.service	3.62 秒	0	0	37	37
com.mooctest.scriptplatform.common	1.67 秒	0	0	17	17
com.mooctest.scriptplatform.utils	2.25 秒	0	0	23	23

图 4.22: 脚本录制与回放项目单元测试报告

4.4.3　功能测试

功能测试通过针对不同模块功能设计测试用例，在一定条件下给定一些输入值及相应的期望输出，判定系统能否按照预期运行并返回结果。通过对预期结果与实际输出结果的比对来发现系统中可能存在的问题进行针对性的缺陷修复，保证跨平台测试脚本录制与回放系统的功能完整性，提高系统稳定性与可靠性。对于系统的功能测试是通过对不同模块的功能进行测试来展开的，用户按照测试用例在系统中实际操作完成对系统功能的验证。

设备管理模块测试主要测试设备管理功能的正确性，包括验证移动设备接入与断开操作时系统能否正常更新设备列表，验证设备使用操作导致设备状态变更时系统能否正常同步设备状态信息到设备列表，验证设备查询操作是否能正常返回系统设备信息列表。

脚本录制模块测试主要测试脚本录制功能的正确性，包括验证选择移动设备操作系统能否正常建立与设备之间的远程连接，验证开始录制脚本操作系统能否正常开始对用户操作进行监听，验证用户点击应用操作系统能否正常记录，验证完成脚本录制操作系统能否正常生成脚本文件。

脚本回放模块测试主要测试脚本回放功能的正确性，包括验证脚本查询操作是否能正常返回数据库中未处于删除状态的脚本记录列表，验证开始脚本回放操作系统能否根据测试脚本，正常完成不同设备上的脚本回放，验证脚本删除操作系统能否正确将脚本记录状态信息更新到数据库中。根据上述各模块功能测试目标，给出系统测试用例套件如表 4.7 示。

表 4.7: 系统测试用例套件

测试用例套件	测试功能	测试内容
TUS1	移动设备接入与断开	接入或断开移动设备，查看系统能否监测到设备变化，更新设备列表
TUS2	移动设备占用与空闲	使用或停止使用移动设备，查看系统能否正常更新设备状态信息
TUS3	查询移动设备列表	进入设备列表页面，查看页面是否显示正确的系统设备信息列表
TUS4	建立移动设备连接	选择操控的移动设备，进入设备远程操控页面，查看是否正常建立与设备的连接
TUS5	开始录制测试脚本	进入脚本录制页面，选择开始录制，查看系统是否对用户操作进行监听
TUS6	用户操作记录	进入脚本录制页面并选择开始录制脚本，用户对设备远程页面进行操控，查看系统是否对操作进行记录
TUS7	完成录制测试脚本	进入脚本录制页面并按照测试用例操控设备进行脚本录制，选择完成录制，查看系统是否正常生成脚本文件
TUS8	查询测试脚本列表	进入脚本记录列表页面，查看页面是否显示正确脚本记录信息列表
TUS9	回放测试脚本	选择回放的设备进入脚本回放页面，选择开始回放脚本，查看页面是否按照脚本记录操作顺序正常回放
TUS10	删除测试脚本	进入脚本记录列表页面，选择删除某条测试脚本，查看脚本列表是否正常更新

根据表 4.7所给出的测试用例套件，从系统的设备管理模块、脚本录制模块与脚本回放模块对功能测试过程及结果分别进行描述。

- 设备管理模块测试

如表 4.8所示，设备管理功能对应测试用例套件包括移动设备接入与断开TUS1、移动设备占用与空闲TUS2、查询移动设备列表TUS3，分别对不同测试套件设计对应的测试用例，以验证设备管理模块功能正确性。经过测试得出测试结果与预期输出结果相符，测试用例全部通过。

表 4.8: 设备管理功能测试用例

ID	说明	输入	预期输出	测试结果
TUS1-1	测试移动设备接入功能	移动设备USB接入系统	移动设备列表出现该接入设备，状态为可用	通过
TUS1-2	测试移动设备断开功能	移动设备USB断开系统	移动设备列表中无该设备信息	通过
TUS2-1	测试移动设备占用状态变更功能	选择使用某空闲移动设备	移动设备列表中对应设备状态为占用，且设备不能被再选用	通过
TUS2-2	测试移动设备空闲状态变更功能	停止使用某移动设备	移动设备列表中对应设备状态为空闲，且设备能够被选用	通过
TUS3-1	测试查询移动设备列表功能	选择查看系统设备列表	页面显示当前系统中所有接入的移动设备信息及状态	通过

- 脚本录制模块测试

如表 4.9所示，脚本录制功能对应测试用例套件包括建立移动设备连接TUS4、开始录制测试脚本TUS5、用户操作记录TUS6、完成录制测试脚本TUS7，针对不同测试套件设计了相应的测试用例，验证脚本录制模块功能正确性。经过测试得出测试结果与预期输出结果相符，测试用例全部通过。

表 4.9: 脚本录制功能测试用例

ID	说明	输入	预期输出	测试结果
TUS4-1	测试建立移动设备连接功能	选择使用某空闲移动设备	跳转设备操控页面，移动设备界面同步显示至操控页面	通过
TUS5-1	测试开始录制测试脚本功能	建立设备连接后，选择开始录制测试脚本	操控页面设备界面区域监听鼠标移动事件，对选中区域红框标识	通过
TUS6-1	测试用户操作记录功能	建立设备连接开始录制后，操控设备界面	操控页面脚本显示区域同步显示用户操作记录	通过
TUS7-1	测试完成录制测试脚本功能	测试脚本录制结束，选择完成录制测试脚本	页面显示脚本录制成功，并显示脚本下载链接	通过

- 脚本回放模块测试

如表 4.10所示，脚本回放功能对应测试用例套件包括查询测试脚本列表TUS8、回放测试脚本TUS9、删除测试脚本TUS10，分别对不同的测试套件设计对应的测试用例，验证脚本回放模块功能正确性。经过测试得出测试结果与预期输出结果相符，测试用例全部通过。

表 4.10: 脚本回放功能测试用例

ID	说明	输入	预期输出	测试结果
TUS8-1	测试查询测试脚本列表功能	选择查看系统测试脚本列表	页面显示当前系统中所有未删除状态的测试脚本记录	通过
TUS9-1	测试回放测试脚本功能	选择回放设备建立连接后，选择回放脚本	操控页面设备界面区域同步脚本执行过程，结束时提示执行成功	通过
TUS10-1	测试删除测试脚本功能	测试脚本列表页面选择删除某脚本记录	系统记录脚本状态，页面提示删除成功并重新加载记录列表	通过

4.4.4　系统运行效果分析

为了进一步验证系统在实际场景下的可用性，本文设计脚本录制与回放实验对系统的运行效果进行分析。由于本系统涉及Android及iOS两种系统类型设备的验证，因此选取5款支持两种系统且分属不同类别的移动应用进行实验。对于脚本回放部分实验，需要对脚本跨平台能力进行验证，同时考虑到市场中Android与iOS设备所占份额差异，选取10台设备作为实验设备集群，包括8台主流品牌Android设备及2台iOS设备。本系统目前支持在Android平台设备上进行脚本录制，通过选取Android 设备与移动应用组合录制测试脚本，验证脚本录制过程对Android设备的支持度。随后从Android设备中选取1台针对5款应用集中进行脚本的录制，并将录制的脚本于设备集群中进行回放，通过对本设备脚本回放成功数与集群中不同设备脚本回放成功数结果进行比较，对系统运行效果进行验证。通过该实验，我们希望验证以下问题。

（1）问题一：本文的系统脚本录制过程对Android平台不同设备的支持度如何？脚本录制是否缩短了脚本开发时间？

（2）问题二：本文的系统脚本回放成功率如何？脚本跨设备、跨平台回放成功率如何？

为对实验提出的问题进行验证，本实验从公司设备中选取10台不同尺寸主流移动设备构建测试设备集群，参考不同系统与不同品牌设备市场占有率选

取华为、三星、小米等主流Android设备8台，iPhone8与iPhone7 Plus主流iOS设备2台。脚本录制与回放实验在该设备集群之上进行展开，设备系统信息及主屏尺寸信息如表 4.11所示。

表 4.11: 实验移动设备信息表

设备编号	设备品牌	系统版本	主屏尺寸(in)
D1	华为	Android8.0	5.1
D2	华为	Android7.0	6.0
D3	三星	Android6.0	5.5
D4	三星	Android8.1	5.7
D5	三星	Android4.4	5.1
D6	小米	Android5.1	5.1
D7	小米	Android6.0	5.5
D8	OPPO	Android7.0	5.5
D9	APPLE	iOS12.1.3	4.7
D10	APPLE	iOS11.3.2	5.5

本实验采用真实移动应用进行测试，考虑到需要在两种系统设备上进行回放，对比Android与iOS应用市场从中选出能够支持两种系统运行且差异较小的5款不同类别、不同复杂度移动应用。通过对该组应用进行脚本录制与回放实验验证上述问题。如表 4.12对应用名称、类别、版本及复杂度给出相应描述。

表 4.12: 待测应用信息表

应用编号	应用名称	应用类别	Android版本	iOS版本	复杂度
A1	小米计算器	工具	1.0.5	1.0.3	低
A2	安兔兔评测	实用工具	7.2.8	7.2.0	中
A3	一个	阅读	4.5.9	4.6.2	中
A4	鲨鱼记账	财务	2.3.7	2.3	高
A5	QQ影音	影音	3.2.0	1.3.2	高

为了对问题一进行验证，从测试设备集群中选取不同品牌Android设备并从提供的5款移动应用中任意选取2款进行组合测试。实验人员选择公司内非测试人员4名，统一提供30分钟了解、熟悉系统使用流程。实验人员通过采用不同品牌Android设备与移动应用组合进行脚本录制，并对录制脚本步骤与时间进行记录。设备与应用组合关系，及录制脚本步骤、所用时间信息如表 4.13所示。

由实验结果可知，本系统能够正常支持测试设备集群中所有Android品牌设备对不同应用进行脚本录制，包括华为、小米、三星及OPPO。对于主流Android品牌以及系统版本4.4 至8.1之间设备具有较好的支持度。根据谷

歌2018发布不同版本系统份额占比[1] 可知，基本实现对90%以上系统版本的覆盖与支持。脚本录制过程对Android平台不同设备良好的支持度为实现脚本跨设备、跨平台回放也提供了相应的保障。同时由不同脚本录制用时与步骤数分析可知，脚本单步录制用时基本保持为10s以下，且录制用时随脚本步骤增加而增加。考虑该实验由非专业人员按照测试用例进行录制操作，结果表明单步录制用时较短，有效降低了测试人员脚本开发技术门槛，能够缩短脚本开发时间。

表 4.13: 待测应用信息表

设备编号	应用编号(步骤数)	用时(s)	应用编号(步骤数)	用时(s)
D1	A1(6)	32	A2(8)	50
D2	A3(7)	41	A4(10)	65
D3	A1(8)	43	A5(9)	56
D4	A2(10)	61	A3(10)	63
D5	A4(11)	71	A5(11)	72
D6	A1(7)	39	A3(9)	57
D7	A2(8)	52	A4(11)	73
D8	A4(10)	55	A5(10)	64

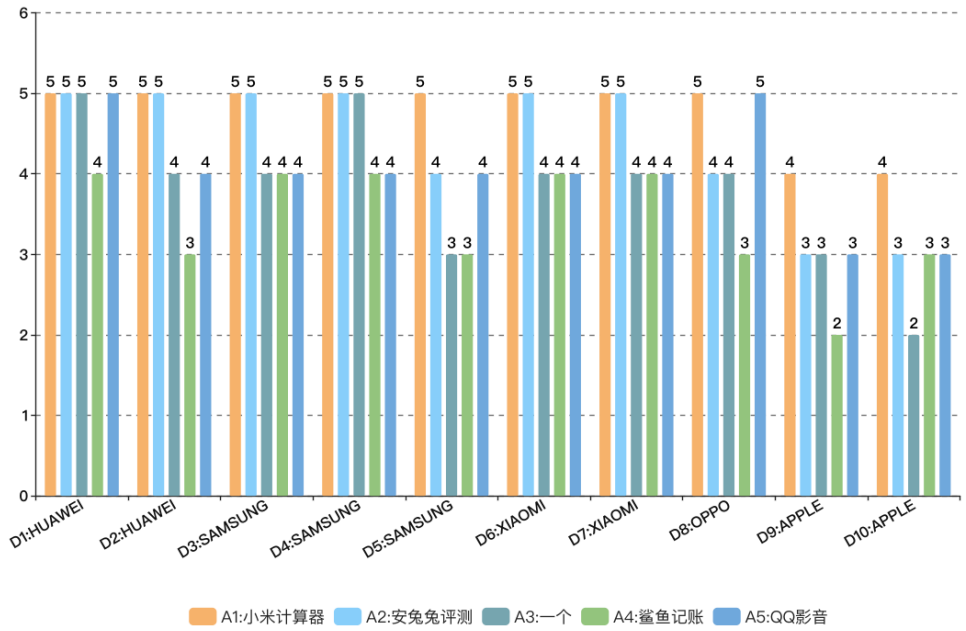

图 4.23: 不同设备脚本回放成功次数统计图

为了验证问题二，本文在编号为D1的华为手机上对选出的5款移动应用分别录制5个测试脚本。之后将录制的25个测试脚本分别于测试设备集群设备上进

[1] https://developer.android.com/about/dashboards/

行回放，脚本回放过程单步回放出现异常不能完全回放脚本记录步骤时认为脚本回放失败，按照设备与应用对脚本回放成功数进行统计。

具体实验结果数据如图 4.23所示，对测试设备集群中每台设备上不同应用的测试脚本回放成功数给出具体数据信息。从图中可知，D1华为作为脚本录制设备，仅在A4应用回放时存在单个脚本回放失败情况，其余应用均能够正常完成脚本的回放，在本设备下脚本的回放成功率达到了95% 以上，保证了系统在回放流程的可用性。A2-A8设备是对脚本跨设备回放进行验证，对回放成功数进行计算可知，Android平台跨设备回放综合成功率在80%以上，设备系统版本及主屏尺寸对应用UI具有一定影响导致了不同设备上回放结果的差异性。A9-A10设备是对脚本跨平台回放进行验证，对回放成功率计算可知，跨平台iOS 设备下回放综合成功率在60% 左右，脚本回放失败主要原因为不同平台应用UI具有一定的差异性，导致定位控件时出现异常。

通过以上实验对本系统运行效果分析可知，本系统支持Android平台下主流设备的脚本录制，录制回放方式有效降低了脚本开发技术门槛，提高了脚本开发效率。在脚本回放时，能够保证在录制设备上正常回放，能够保证脚本跨设备回放时具有较高成功率，同时能够支持脚本跨平台回放，在两平台应用UI差异较小时具有较高的回放成功率。实验结果表明，本系统提高了测试脚本的跨设备、跨平台能力，具有良好的可用性。

4.4.5　系统运行展示

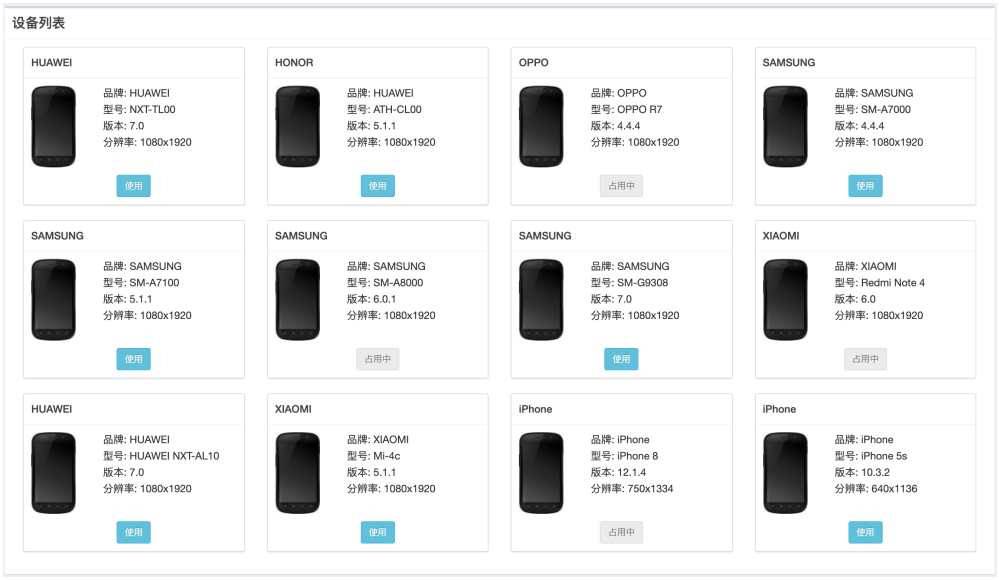

图 4.24: 设备列表界面运行截图

设备列表界面如 4.24 所示，测试人员可以通过设备列表界面查看当前系统中连接设备的信息，包括设备品牌、型号及系统版本等。同时可以选择系统中状态为可使用的设备建立远程连接进行脚本录制回放。当系统中设备被占用时对应按钮提示文字变为占用中，按钮不可点击，提示该设备不能被重复选用。

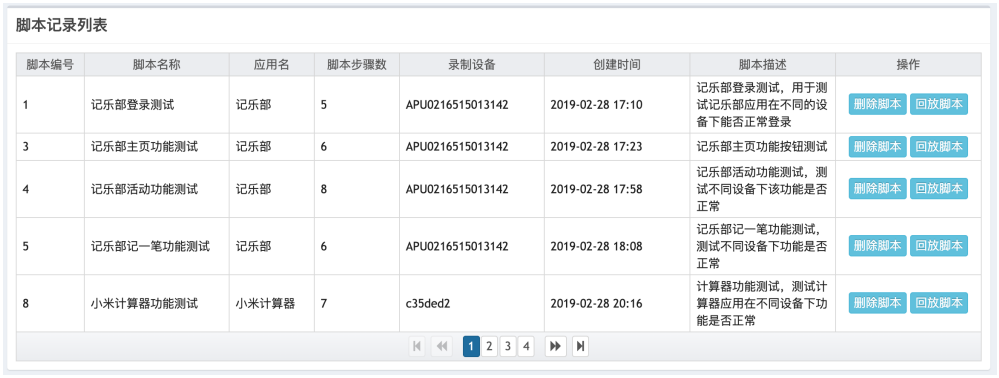

图 4.25: 脚本记录列表界面运行截图

如图 4.25所示为脚本记录列表界面。测试人员通过脚本记录列表可以查看录制完成的脚本记录信息，包括脚本编号、名称、步骤数与应用名及录制设备等。脚本记录对应操作列包含查看脚本与回放脚本按钮，可以实现对脚本记录删除和回放。点击脚本回放按钮，会跳转设备列表界面进行回放设备的选择，设备选定后正常建立设备连接，能够对录制脚本进行回放。

图 4.26: 脚本录制界面运行截图

脚本录制页面如图 4.26所示，主要包括设备交互区域与测试脚本展示区域两个组成部分。设备交互区域主要将远程设备界面同步展示给测试人员，同时提供对操控事件的监听，允许测试人员进行设备的远程操控。该区域主要负责了测试人员脚本录制过程中与设备间的交互。

测试脚本展示区域包括脚本录制流程控制按钮、下载脚本按钮及脚本步骤展示部分组成。通过开始录制按钮启动设备交互区域对测试人员操作的监控，以红色边框标示当前选中控件，完成录制用于结束脚本录制流程，下载按钮提供录制脚本下载服务。测试脚本记录则以操作和控件截图组合形式在展示区域实时对测试人员进行展示。

图 4.27: 脚本回放界面运行截图

图 4.27给出了脚本回放界面，为培养用户使用习惯，该界面与脚本录制界面采用统一布局，仅对脚本展示区域功能按钮进行替换。测试人员可以通过点击开始回放按钮开始脚本的自动回放，具体脚本记录可在展示区域进行查看，脚本实时回放过程在设备交互区域实时同步展示。切换设备按钮方便测试人员在当前页面进行不同设备的切换，下载脚本按钮即提供测试脚本下载服务。

4.5　本章小结

本章主要介绍了跨平台脚本录制与回放系统设备管理模块、脚本录制模块及脚本回放模块，通过顺序图、主要类表及类图对各模块主要功能的详细设计

与实现进行说明，并给出了关键函数代码展示了模块功能内部的具体实现细节。最后对系统部署并进行单元测试和功能测试，通过设计实验对系统运行效果进行分析，展示系统运行时界面截图并给出相关说明。

第五章　总结与展望

5.1　总结

随着移动互联网的飞速发展，越来越多的用户不再满足于对移动应用起步时的功能方面的需求，转而对于应用软件的质量要求逐渐提升。软件测试对于确保软件的质量至关重要，而GUI测试又是移动应用软件测试的关键、主要组成部分。目前企业较为常见GUI测试解决方案是测试人员根据业务流程编写测试脚本完成自动化测试。但由于移动应用往往运行于多系统、多设备，测试脚本跨平台、跨设备运行的局限性导致测试脚本需要针对不同系统、设备进行重复性开发与维护，且测试脚本的开发与维护往往需要测试人员具备一定的编程水平，使得测试门槛进一步提高。为了解决GUI测试中测试脚本跨平台、跨设备能力不足导致脚本重复开发、维护带来的自动化测试工作困难、效率低的问题，同时改进脚本开发方式为录制与回放，简化了测试流程，降低测试门槛要求，因此设计实现了基于图像识别的跨平台测试脚本录制与回放系统。

本文首先介绍了移动应用自动化测试应用的背景及GUI自动化测试中存在的问题，然后对较为常见的自动化测试框架发展现状进行了对比分析，之后对系统中使用到的SIFT特征匹配算法，WebSocket协议，Tesseract-OCR技术，WDA工具和ADB工具及Netty框架进行了相关介绍。

本系统为了降低测试脚本开发、维护的技术门槛，基于远程设备操控，采用录制与回放形式进行脚本开发，便捷、精准的录制方法降低了脚本开发的复杂性以及测试人员的人工成本。为了提升测试脚本的跨平台、跨设备执行能力，本系统引入控件截图、布局位置信息作为控件属性，图像处理模块基于截图与布局位置信息实现了控件在不同系统与设备上的定位，进而实现脚本单次录制多机型、跨平台回放，减少了企业维护脚本数量。

在技术设计方面，使用SpringBoot加速项目的搭建与开发，利用Netty框架与WebSocket协议构建移动设备与浏览器之间信息实时传输通道，使用MiniCap工具实现iOS与Android设备界面在浏览器端同步展示，使用SIFT与Tesseract OCR构建图像匹配与布局匹配服务，保障脚本跨平台回放的稳定性，可靠性。系统模块间以微服务形式进行交互，降低了模块的耦合性，提高了系统的扩展性与可维护性。

系统目前已在公司内部部署供测试人员使用。系统一定程度上提升了移动应用GUI测试的效率，提高了测试脚本的跨平台、跨设备能力，降低了测试人

员脚本开发技术门槛，减少了移动应用需要维护的测试脚本数量，缩短了应用开发迭代周期，为公司移动应用测试节省了成本。

5.2　工作展望

跨平台测试脚本录制与回放系统已在公司内部内测使用，但由于系统涉及移动应用测试的多个系统平台、不同的移动设备，由于移动应用类型、布局、设备、平台等因素带来的复杂性以及研发时间的限制，本文设计实现的系统仍存在一些不足之处，未来可以从以下方面来进行系统的优化与改进。

(1) 本系统主要着手于测试流程方面的改进并对移动设备涉及的主要操作进行覆盖，在后期系统迭代完善时可以增加对移动设备不同操作的封装，实现对移动设备更多操作类型录制与回放的支持。

(2) 目前系统已实现对Android设备的录制回放及iOS设备的回放过程的支持。系统设计实现时考虑到对移动设备及平台的扩展需求，对设备管理抽象化设计，提供了统一的扩展接口，方便不同平台设备进行系统接入。系统迭代时可以对iOS测试工具进一步研究，增加对iOS设备录制支持，同时可以根据市场需求对不同平台系统进行覆盖支持。

(3) 系统核心功能为实现测试脚本录制与跨平台回放，在测试生态体系方面本系统仍有较大提升空间。如增加测试报告管理系统，可以通过记录脚本回放过程中设备运行数据，增加对回放过程中日志与数据的汇总、分类和处理，根据企业自身需求对数据处理结果构建测试报告并提供报告管理服务，完善测试生态体系。

(4) 随着图像处理、机器学习及人工智能等技术的发展，可以对目前系统中使用到的图像处理技术进行优化改进，提升匹配精度与识别速度。同时，可以尝试借助机器学习、人工智能等技术对控件图片进行分类处理，引入控件类别信息来进一步辅助控件定位，提升系统跨平台回放稳定性。

参考文献

[1] 中商产业研究院, 2018年中国移动互联网市场分析及预测, www.askci.com/news/chanye/20180912/1554451131848.shtml (2018).

[2] K. Li, M. Wu, Effective GUI Testing Automation: Developing An Automated GUI Testing Tool, John Wiley & Sons, 2006.

[3] M. Fazzini, Automated Support for Mobile Application Testing and Maintenance, in: Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, pp. 932–935.

[4] A. M. Memon, GUI Testing: Pitfalls and Process, IEEE Computer 35 (8) (2002) 87–88.

[5] T. Grønli, G. Ghinea, Meeting Quality Standards for Mobile Application Development in Businesses: A Framework for Cross-Platform Testing, in: Proceedings of the 49th Hawaii International Conference on System Sciences, 2016, pp. 5711–5720.

[6] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. D. Carmine, G. Imparato, A Toolset for GUI Testing of Android Applications, in: Proceedings of the 28th IEEE International Conference on Software Maintenance, 2012, pp. 650–653.

[7] H. Song, S. Ryoo, J. H. Kim, An Integrated Test Automation Framework for Testing on Heterogeneous Mobile Platforms, in: Proceedings of the 1st ACIS International Symposium on Software and Network Engineering, 2011, pp. 141–145.

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

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

[10] S. R. Choudhary, Cross-platform Testing and Maintenance of Web and Mobile Applications, in: Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 642–645.

[11] 何梁伟, 基于AI的移动端自动化测试框架的设计与实现, 技术报告, 爱奇艺测试部(2018).

[12] T. A. Nguyen, C. Csallner, Reverse Engineering Mobile Application User Interfaces with REMAUI , in: Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering, 2015, pp. 248–259.

[13] T. Yeh, T. Chang, R. C. Miller, Sikuli: Using GUI Screenshots for Search and Automation, in: Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, 2009, pp. 183–192.

[14] J. Sun, S. Zhang, S. Huang, Z. Hui, Design and Application of a Sikuli Based Capture-Replay Tool, in: Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion, 2018, pp. 42–44.

[15] J. A. Whittaker, What Is Software Testing? Why Is It So Hard? Practice Tutorial, IEEE Software 17 (1) (2000) 70–79.

[16] R. P. Testardi, Methods and Systems for Automated Software Testing, US Patent 6,249,882 (2001).

[17] A. M. Memon, M. E. Pollack, M. L. Soffa, Automated Test Oracles for GUIs, in: Proceedings of the 8th ACM SIGSOFT International Symposium on Foundations of Software Engineering: Twenty-first Century Applications, 2000, pp. 30–39.

[18] L. Zhifang, L. Bin, G. Xiaopeng, Test Automation on Mobile Device, in: Proceedings of the 5th Workshop on Automation of Software Test, 2010, pp. 1–7.

[19] S. Amatya, A. Kurti, Cross-platform mobile development: Challenges and opportunities, in: Proceedings of the 2013 ICT Innovations and Education, 2013, pp. 219–229.

[20] S. Xanthopoulos, S. Xinogalos, A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications, in: Proceedings of the 6th Balkan Conference in Informatics, 2013, pp. 213–220.

[21] S. Hwang, H. Chae, Design & Implementation of Mobile GUI Testing Tool, in: Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology, 2008, pp. 704–707.

[22] D. Hayes, Mobile Web App Development for All!: (Abstract Only), in: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 2018, pp. 1060–1060.

[23] I. Malavolta, S. Ruberto, T. Soru, V. Terragni, Hybrid Mobile Apps in the Google Play Store: An Exploratory Investigation, in: Proceedings of the 2nd ACM International Conference on Mobile Software Engineering and Systems, 2015, pp. 56–59.

[24] 侯津，顾乃杰，丁世举，杜云开, 基于控件路径的跨设备UI自动化测试方法, 计算机系统应用27 (10) (2018) 240–247.

[25] M. Mercieca, WebDriverAgent:Getting Started With Automated iOS Testing, www.mutuallyhuman.com/blog/2017/04/20/webdriveragent-getting-started-with-automated-ios-testing (2017).

[26] S. Hwang, S. Lee, Y. Kim, S. Ryu, Bittersweet ADB: Attacks and Defenses, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, 2015, pp. 579–584.

[27] R. Regupathy, Android Debug Bridge (ADB), Apress, Berkeley, CA, 2014.

[28] J. Amarante, J. P. Barros, Exploring USB Connection Vulnerabilities on Android Devices - Breaches Using the Android Debug Bridge, in: Proceedings of the 14th International Joint Conference on e-Business and Telecommunications, 2017, pp. 572–577.

[29] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[30] A. Bruno, L. Greco, M. L. Cascia, Object Recognition and Modeling Using SIFT Features, in: Proceedings of the 2013 International Conference on Advanced Concepts for Intelligent Vision Systems, 2013, pp. 250–261.

[31] E. Karami, S. Prasad, M. S. Shehata, Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images, arXiv preprint arXiv:1710.02726.

[32] R. Smith, An Overview of the Tesseract OCR Engine, in: Proceedings of the 9th International Conference on Document Analysis and Recognition, 2007, pp. 629–633.

[33] I. Marosi, Industrial OCR Approaches: Architecture, Algorithms, and Adaptation Techniques, in: Document Recognition and Retrieval XIV, San Jose, California, USA, 2007., 2007, p. 650002.

[34] 李林峰, Netty权威指南, 北京：电子工业出版社, 2014.

[35] C. Yuan, D. M. Li, Y. Li, Review on Real-Time Communications Technology in Web Application, Advanced Materials Research 1044-1045 (2014) 1309–1314.

[36] A. Lombardi, WebSocket: Lightweight Client-Server Communications, O'Reilly Media, Inc., 2015.

[37] V. Pimentel, B. G. Nickerson, Communicating and Displaying Real-Time Data with WebSocket, IEEE Internet Computing 16 (4) (2012) 45–53.

[38] T. Chang, T. Yeh, R. C. Miller, GUI testing using computer vision, in: Proceedings of the 28th International Conference on Human Factors in Computing Systems, 2010, pp. 1535–1544.

简历与科研成果

基本情况 赵文远，男，汉族，1995 年 1 月出生，河南省郸城县人。

教育背景

2017.9～2019.6 南京大学软件学院 硕士

2013.9～2017.6 华中科技大学软件学院 本科

参与项目

1. 南京南瑞集团项目：移动众测平台软件（20170413），2017-2018
2. 江苏通行宝智慧交通科技有限公司：电子账户、MTC电子支付和苏卡通客户行为分析+软件测试项目，2018-2019
3. 国家自然科学基金项目（面上项目）：协作式众包测试报告分析与融合技术研究（61772014），2018-2021

致　　谢

时光如白驹过隙，总是来不及珍惜便匆匆地从身边飞逝。很快我即将结束为期两年的研究生生活，在这两年的研究生学习、工作时光中，我得到了许多师长、同学、朋友的关怀和帮助。在论文即将完稿之际，我想对所有曾经给过我关怀和帮助的人表达最诚挚的感谢!

首先我要感谢的是我的导师陈振宇教授、房春荣老师。在研究生阶段的两年时间里，陈老师为我们提供良好的学习、工作环境，同时在科研方面给了我很多帮助与指导。在毕设阶段，陈老师、房老师从开始选题、立项答辩到后期项目的设计与实现过程中都给予了我很大的帮助，提出了很有实质性的意见建议，让我能够及时发现项目中的问题并做出改正。在他们的悉心指导下，最终能够顺利的完成此项目及论文。同时陈老师、房老师在生活中严于律己、一丝不苟的治学精神也带给了我巨大的影响，督促着我在离开校园，走向以后的人生道路中仍要坚持严于律己，不断学习，努力奋斗。

其次我要感谢研究生期间与我一起学习、工作的“iSE 实验室”的同学们，两年的时光我们一同学习、成长，从他们身上我不仅学到了技术方面的知识，也学到了努力、乐观、积极的生活态度。

特别地，我要感谢我的家人和女友，是他们坚定不移的支持着我的求学道路，他们的关怀、鼓励与陪伴，是我能够走到今天的最大动力。

最后对参加答辩的各位专家、老师们能在百忙之中抽出时间审阅我的论文表示衷心的感谢!

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