

本科生毕业论文（设计）



题 目  **长桥停车**

学 院  **计算机学院**

专 业  **计算机科学与技术**

学生姓名  **李治成**

学 号  **2014141462126** 年级 **2014**

指导教师  **潘薇**

教务处制表

二Ο一七年五月十五日

**微信智能车库**

专业 计算机科学与技术

学生 李治成 指导老师 潘薇

**[摘要]** 近几十年来，我国盲目快速推动城市化进程，而各地的城市化进程过程中，却没有相应的在城市基础设施建设以及城市管理方面加强改进，停车位建设远远赶不上汽车增长的速度。依托互联网平台，民间资本的涌动加上国家政策支持，停车行业迎来了属于自己的春天，出现的智能车库平台也发展比较快，已出现了很多移动app管理车库平台，但是对于微信这个新宠，有关智能车库的公众号或者微信小程序少之又少，满足不了大多数客户的需求。

因此，本项目是基于智能停车产业界出现的现实用户体验问题提出来的，旨在通过分析研究智能停车人机交互用户体验设计，在一定程度上缓解和优化现有的问题，让众多停车用户能够对基于移动互联网和车联网技术的停车系统产生信任和依赖，成为他们的停车好帮手

**[主题词]** 微信公众号；停车； 优化

**Design and Implementation of Curriculum Schedule Application Based on Android**

Major : Computer Science and Technology

Student : Yuan Yuxin Adviser : Pan Wei

[Abstract] Recent years, the public is paying more attention to the time management. As college students, it is significant to learn how to balance the time between study and activities after school while our campus life is increasingly colorful. Therefore, this paper designs and implements an application called *ChuanDaYouYue*, aiming at the students in Sichuan University, in order to help students plan the after-class activities and improve the efficiency. This application optimizes the function of the student curriculum schedule, shows the curriculum and spare-time plans in the form of timetable which combines the school lessons with the after-class planning, and provides convenient operations and reminder functions. At the same time, the plan can be released into an invitation, which could invite your partners to join together to improve the efficiency of tasks. This application designs and implements the client in Android and server in J2EE respectively, realizing the data transmission with JSON format through the Internet. The client can upload the data to server to process and synchronize the processed data in server so that the user can use the application in different Android devices. In addition, it can import courses through The Office of Academic Affairs using Web Crawler, which is convenient to the students to manage the curriculum. Finally, this application is deployed on the testing simulator, implementing the function of curriculum management, personal schedule planning, invitation management and reminders and mute management, which can well meet the needs of users.

[Keyword] Android; Curriculum; plan schedule; invitation; reminder

目录

[第一章 绪论 1](#_Toc483347318)

[1.1 项目背景 1](#_Toc483347319)

[1.1.1 大学生在校时间利用情况分析 1](#_Toc483347320)

[1.1.2 目前主要课表类应用的比较和分析 1](#_Toc483347321)

[1.1.3 本项目的研究意义 2](#_Toc483347322)

[1.2 本文的主要工作和结构框架 2](#_Toc483347323)

[1.2.1 论文的主要工作 2](#_Toc483347324)

[1.2.2 论文的结构框架 3](#_Toc483347325)

[第二章 相关技术介绍与分析 4](#_Toc483347326)

[2.1 Android客户端 4](#_Toc483347327)

[2.1.1 Android应用程序组件 4](#_Toc483347328)

[2.1.2 SQLite数据库 4](#_Toc483347329)

[2.2.3 JSON数据传输格式 4](#_Toc483347330)

[2.2 Android开源库 5](#_Toc483347331)

[2.2.1 OkHttp开源框架 5](#_Toc483347332)

[2.2.2 GSON开源库 5](#_Toc483347333)

[2.2.3 jsoup解析库 5](#_Toc483347334)

[2.3 服务器端框架 5](#_Toc483347335)

[2.4 本章小结 6](#_Toc483347336)

[第三章 应用需求分析 7](#_Toc483347337)

[3.1 功能性需求分析 7](#_Toc483347338)

[3.2 非功能性需求分析 8](#_Toc483347339)

[3.3 应用设计目标 8](#_Toc483347340)

[3.4 本章小结 8](#_Toc483347341)

[第四章 应用概要设计 9](#_Toc483347342)

[4.1 应用功能模块 9](#_Toc483347343)

[4.1.1 课程管理模块 11](#_Toc483347344)

[4.1.2 个人规划管理模块 11](#_Toc483347345)

[4.1.3 邀约管理模块 11](#_Toc483347346)

[4.1.4 提醒与上课静音模块 12](#_Toc483347347)

[4.1.5 登录注册模块 12](#_Toc483347348)

[4.1.6 设置模块 12](#_Toc483347349)

[4.2 客户端总体架构 13](#_Toc483347350)

[4.3 服务器端架构与数据库设计 14](#_Toc483347351)

[4.3.1 框架与响应流程 14](#_Toc483347352)

[4.3.2 数据库设计 14](#_Toc483347353)

[4.4本章小结 17](#_Toc483347354)

[第五章 应用详细设计和实现 18](#_Toc483347355)

[5.1 客户端的详细设计与实现 18](#_Toc483347356)

[5.1.1 滑动菜单的设计与实现 19](#_Toc483347357)

[5.1.2 动态显示主界面的设计与实现 20](#_Toc483347358)

[5.1.3 爬虫的设计与实现 20](#_Toc483347359)

[5.1.4 课程管理模块 20](#_Toc483347360)

[5.1.5 个人规划管理模块 21](#_Toc483347361)

[5.1.6 邀约模块 21](#_Toc483347362)

[5.1.7 提醒与上课静音模块 22](#_Toc483347363)

[5.1.8 登录注册模块 22](#_Toc483347364)

[5.1.9 设置模块 23](#_Toc483347365)

[5.2 服务器端的详细设计 23](#_Toc483347366)

[5.2.1 控制层 24](#_Toc483347367)

[5.2.2 数据处理层 25](#_Toc483347368)

[5.3 本章小结 26](#_Toc483347369)

[第六章 测试与成果展示 27](#_Toc483347370)

[6.1 测试环境 27](#_Toc483347371)

[6.2 功能性测试 27](#_Toc483347372)

[6.3 兼容性与性能测试 28](#_Toc483347373)

[6.4 成果展示 28](#_Toc483347374)

[6.4.1 用户注册登录 29](#_Toc483347375)

[6.4.2 课程管理 29](#_Toc483347376)

[6.4.3 个人规划管理 31](#_Toc483347377)

[6.4.4 邀约管理 31](#_Toc483347378)

[6.4.5 提醒与上课静音 33](#_Toc483347379)

[6.4.6 设置 34](#_Toc483347380)

[6.5 本章小结 34](#_Toc483347381)

[第七章 总结 35](#_Toc483347382)

[参考文献 36](#_Toc483347383)

[声 明 37](#_Toc483347384)

[致 谢 38](#_Toc483347385)

[附录1 39](#_Toc483347386)

[附录2 45](#_Toc483347387)

# 第一章 绪论

## 1.1 项目背景

随着我国城市经济和汽车工业的迅速发展，拥有私家车的家庭越来越多，而与此相对应的是城市停车状况的尴尬，特别是大城市的机动车拥有量的增长速度远远超过停车基础设施的增长速度，泊位与车数量的比例严重失调，由此带来停车难、违章停车、停车贵、停车管理困难等一系列问题日益突出，轿车露宿街头的景象在许多城市随处可见。解决好出行问题已经成为城市发展的重点之一，我们必须重视城市停车难的问题，并积极探求解决的措施。

智能停车系统建设已经纳入智慧城市建设板块，成为具有经营性的城市智能交通项目，未来，城市停车将逐步进入智能时代，通过智能停车方案解决人工管理、施工难度大、运维复杂、车主体验差、收费管理混乱等问题，在这个阶段智能停车方案将深入到城市每个角落，影响每个车主和停车运营厂商。

微信公众平台简介

微信，是时下最受关注的移动互联网产品。经过长时间的发展，微信的注册已经超过8亿，成为移动互联网最重要的入口之一，微信公众平台是腾讯公司在微信基础上新增的功能模块，通过这一个平台，个人与企业都可以打造一个微信公众平台，并实现与特定群体的文字、拖以及语音的全方位沟通与交流。公众平台主要面向名人、政府、媒体、企业等，在这里可以通过微信渠道将品牌推广给上亿的微信用户，减少宣传成本，提高品牌知名度，打造更具影响力的品牌形象。

### 1.1.1传统停车场利用情况分析

传统停车场仍占市场主导，目前传统停车场建设规模仍占接近90%的市场份额，新型的机械式停车场的建设虽一片火热，但整体市场占比仅10%左右，传统的停车场管理系统只解决了出入口控制的问题，对于停车场停车信息，停车场实时数据等功能望尘莫及，而且在收费这个环节上也存在缴费方式单一、人工管理效率低下、存在收费漏洞等问题，更别说进行停车场整体的系统整合及资源优化配置了。目前国内的停车场尤其是大厦的停车场，收费管理部分做得尤为完善，但是对于一些小型停车场、私有停车场的应用管理，功能比较完善的，也是基于APP应用开发的，但是对于微信平台，公众号与小程序开发出来的功能太少，要么界面太丑，应用信息不够完善，用户宁愿用APP而不用微信平台提供的服务。

### 1.1.2 目前市场上移动智能停车场软件应用的比较和分析

目前，应用市场上关于此类型的软件鱼龙混杂，本文选取国内外比较受欢迎的三款app应用PP停车、ETCP停车、宜停车加以分析。这四款软件都是针对停车场开发的软件，能够实现停车场查找、管理、车位市场发布等功能，对比其最基础的功能可以发现可以有以下这些功能：最基本的功能是定位当前位置、实时查看周边停车位数量及停车费用、导航到指定停车位、手机支付停车费用（提供停车券活动）、分享app等功能。这三款软件早早地进入了市场，得到了一些投资者的青睐，而且积累了一定量的停车场客户。有较多的收费停车场的客户，就会对停车场的业务相比其他后来者要熟练不少。其次，在线支付功能易用，支持微信、支付宝支付，而且可以直接充值到app。有良好的业务能力，足以满足一般停车需求，不过这些应用软件数据库还并不是很大。

目前国内停车位APP服务的比较多，对于微信公众号少之又少。app停车主要侧重在闲置车位出租上，ETC停车场主要侧重找车位，还有的一些停车APP则是什么违章查询之类的，但是对于一些小型停车场、私有停车场的应用管理，功能比较完善的，也是基于APP应用开发的，但是对于微信平台，有较大的开发利用价值。

### 1.1.3 国内的停车发展现状分析以及未来发展

国内的停车应用比较多了，ETCP、好停车、吾舍停车等，但是一直做不起来，具体分析有以下几个原因：

1.就目前市场上的应用来看，目前的难点在线下车位渠道上，缺少相应的政策支持，大部分物业、商圈、市政是不愿意把车位拿出来贡献的，需要找到利益切合点才好开展工作。

2. 产业联调太长，如果和小区打通，涉及物业、闸机厂商、业主委员会，再加上提供车位的人，利益分配难统一，谈判成本过高；

3. 闸机设备的厂商太多太杂，要和这么多的厂商打通不容易，且有的用的是车牌识别，有的是蓝牙卡，门槛不高，各个地方又有很多的小厂商。

4. 目前的车位共享其实更多的是租赁经济，一个车位运营商去和停车场合作，把车位拿过来再租给需要车位的人，并不是真正的C2C共享，少了一方的参与者，是完全的重资产模式，需要不断的融资去拿车位，市场不容易爆发。

智能车库发展方向：

一是将更多的高科技使用到停车设备中，韩国和美国的车库行业将机电技术成果随时转化和移植到车库产品中，使车库产品更新很快，也使车库的存取车速度加快，存车量变大，从而逐步代替老式停车设备。

二是越来越重视停车设备的性价比和经济性。在保证停车设备的性能质量前提下，生产厂家更重视如何提高产品的实用性和降低单车位造价。

### 1.1.4 本项目的研究意义

停车场智能化管理的现状将成为未来的发展趋势，随着我国劳动力成本的快速上升，停车场的管理人员越来越少，自动化程度越来越高，无人化服务将逐渐普及，所以，安装[智能化](http://link.zhihu.com/?target=http%3A//www.sztxsy.net/a/zhinenhuaxiliechanpin/)[停车场管理系统](http://link.zhihu.com/?target=http%3A//www.sztxsy.net/index.html)将成为未来大型停车场的一种趋势。

随着智能停车场的发展，很多地方的停车场将实现联网共享数据，可通过场内停车量、停车收费额、客流方向、等数据了解车主们的需求，目的是使人们停车缴费更方便。由于现在停车难和乱收费问题已是人们最为头疼的问题，为了解决这一问题，停车场管理将逐步走向智能化。停车场智能化管理主要体现在停车场安装的智能停车场管理系统，停车场管理系统包括出入口管理、车位引导、反向寻车、车牌自动识别、自助缴费等很多个子系统组成。同时采用中央收费模式、远距离识别技术等来实现智能化管理的功能。在一些大型的停车场内我们可以看到人们从停车到缴费整个过程只需短短的几分钟时间即可完成，不再像之前那样，把大部分时间都浪费在停车上面。

其次停车场的发展趋势将逐步走向智能化、无人化。所以传统的停车场管理已不能满足人们的需求，很多地方的停车场开始进行改造，升级为车牌识别系统，同时停车场内安装车位引导和反向寻车及自助缴费系统，这样进出停车场的车辆可以通过车位引导系统快速的找到停车位，返回停车场时，由于场内车辆较多，一时间不能找到自己爱车停放的位置，通过反向寻车系统，车主只需几分钟时间就可找到车辆的存放位置，同样缴费方式也比之前更加的多样化，人们可以选择在停车场的出口处通过停车场收费系统的提示进行缴费，也可以通过场内的自主缴费机缴费，这样不但不会出现乱收费现象，还可以节省车主排队缴费的时间。

微信公众平台，作为微信最重要的部分，以“再小的平台也有自己的平台为宣传口号”，吸引了众多个人与企业加入，争先在微信平台上推广自己，到目前为止，每年都在以80000个左右的速度增长，微信公众平台经过这么久的发展，受到了广大开发者的青睐。

## 1.2 本文的主要工作和结构框架

### 1.2.1 论文的主要工作

本文主要介绍系统的功能、实现、性能以及相关技术。开发此公众号主要是从目前市场出发，借助微信强大的连接能力，将停车系统、商场、服务及用户连通互动，实现online和offline的真正无缝融合。下面是关于本项目的一些介绍：

1）选用公众号，而不是传统的APP

传统的APP，不仅占手机内存，还需要损耗手机手机流量，增加了用户的抵触情绪，支付功能一般都是第三方支付，需要填写很多资料，比较繁琐复杂，启用微信公众号，一切就变得更加简单，公众号有推送功能，界面简洁，易于操作，便捷支付，这个方面来说比传统的APP方便。目前在国内，公众号是最佳的吸粉神器，通过公众号车主可以很快速的使用相应服务，公众号有天然的优势，可以有很多的信息推送以及增值服务可供选择，后期的维护更加方便。

2）实时获取周围停车场，提供到停车场的路径导航

相比传统的微信公众号而言，根据定位情况，加载出来周围所有停车场，网页的限制，导航去停车场往往只能调用第三方，而本公众号，提供了路径导航，根据终点，提供去此终点的路径导航，用户可以根据自己的情况选取路线。

3）更加注重系统性能

比起传统的项目，完全不考虑数据库的性能，晚期只要涉及到数据库的增、删、改、查的时候完全不考虑数据库的压力，在项目中，应用了hibernate的一些性能对数据库进行了优化，减少了对数据库的操作，很大程度上提高了系统的性能，在用户量很大的情况下，依旧可以保持良好的运行。

基于以上三点的设计方案，本论文开展的主要工作有：

1. 学习微信公众号开发的方法，了解微信的相关的技术，搭建微信平台；
2. 利用百度地图提供的接口，解决停车场位置、定位问题。
3. 项目主要利用java语言编写，采用ssh（spring，springMVC，hibernate）框架，最大程度上优化系统性能。

4）采用瀑布模型开发方法，从用户需求分析开始，通过策划、建模、构建与部署的过程，最终提供一个完整的软件。

### 1.2.2 论文的结构框架

第一章绪论介绍了项目背景，当前国内外停车场的发展状况，分析国内停车场发展不起来的原因，阐述微信公众号优点，项目的开发意义。

第二章介绍开发微信公众号相关的技术。

第三章通过用例图分析该智能车库应用的用户需求，描述本应用的功能并提出应用设计目标

第四章采用UML建模，通过需求用例分析对应用的总体架构进行概要设计

第五章阐述微信智能车库应用的详细设计以及实现，包括系统性能设计，界面设计以及类的设计。

第六章介绍了应用测试，列出了主要的测试用例和测试结果

第七章对本项目进行了总结，介绍了应用存在的不足以及发展方向。

# 第二章 微信公众号的技术分析

本公众号开发主要利用微信提供的API进行开发，因为接口限制，申请微信测试号来解决接口问题。以eclipse为开发工具，利用Apache tomcat v9.0搭建服务器。客户端利用javascript、jquery、css、html、bootstrap开发的响应式页面，优化界面，服务端利用java语言编写，采用ssh（spring、springMVC、hibernate）框架简化开发流程。利用mysql来保存用户的数据。客户端与服务器交互采用json数据，springMVC提供的模型、视图、控制层的开发，方便了界面与服务器的交互。

## 2.1 微信公众平台介绍

现如今，微信已经不再只承担着交流沟通、娱乐大众的功能，微信公众号的推出将微信逐渐转变成个人、商家、企业单位用来营销的重要工具。对于公众号平台的开发，官方只给出了PHP示例代码，本项目使用的是java语言开发。公众平台的处理程序其实就是一个web项目，负责接收并相应微信服务器发送的http请求。只要能支持动态web开发的编程语言都能用于公众号开发。

## ssh框架介绍

SSH框架一般指的是Struts、Spring、Hibernate，后来Struts2代替了Struts，最近5年，Struts2已经被Spring MVC代替，而Hibernate基本也被iBatis/MyBatis代替。本项目采用的是spring，springMVC,hibernate的框架。使用框架的目的提高程序的可读性和维护性，框架的核心思想是按照关注点分离，按照MVC的框架进行分离。

### 2.1.1 spring技术

Spring基于IOC和AOP的结构J2EE系统的轻量级的开发框架，框架的基础版本在2 MB 左右，使用 Spring 框架可以创建好性能、易于测试、可重用的代码，为容易而快速的开发出耐用的 Java 应用程序提供了全面的基础设施。

Spring涉及到相关技术：

IOC（Inversion Of Control）反转控制是Spring的基础，对象的生命周期由Spring来管理，直接使用Spring去获取对象，创建对象由以前的程序员自己构造变成了交由Spring创建。

DI（Dependency Inject）依赖注入.拿到的对象的属性，已经被注入好相关值，可以直接使用。

AOP 即Aspect Oriented Program面向切面编程，能够选择性的、低耦合的把切面和核心业务功能结合在一起的编程思想。在面向切面编程的思想里面，把功能分为核心业务功能和周边功能。所谓的核心业务，比如登陆，增加数据，删除数据都叫核心业务。性能统计，日志，事务管理等等是周边业务，周边业务也被定义为切面。

### 2.1.2 springmvc

Spring Web MVC是一种基于Java的实现Web MVC设计模式的请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将web层进行职责解耦，基于请求驱动指的就是使用请求-响应模型。SpringMVC是Spring框架的一个模块，SpringMVC和Spring无需通过中间整合层进行整合。Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离更容易进行定制。

Spring 的 Web MVC 框架是围绕DispatcherServlet进行设计，把请求分派给处理程序，同时带有可配置的处理程序映射、视图解析、本地语言、主题解析以及上载文件支持。应用控制器其实拆为处理器映射器(Handler Mapping)进行处理器管理和视图解析器(View Resolver)进行视图管理，页面控制器是非常简单的Controller接口，只有一个方法ModelAndView handleRequest(request, response)。Spring 为其提供一个控制器层次结构，可以派生子类。

### 2.2.3 hibernate

使用JDBC做数据库相关功能开发会做很多重复性的工作，比如创建连接，关闭连接，把字段逐一映射到属性中。Hibernate是数据访问层的框架，对JDBC进行了封装，是真对数据库访问提出的面向对象的解决方案。使用Hibernate可以直接访问对象，自动将次访问转化成SQL执行，从而达到简介访问数据库的目的，简化数据访问层的代码开发。

## 2.2 前端技术开发

### 2.2. javascript

JavaScript是世界上最流行的脚本语言，在电脑、手机、平板上浏览的所有的网页，以及无数基于HTML5的手机App，交互逻辑都是由JavaScript驱动的。JavaScript依赖于浏览器本身，与操作环境无关，只要计算机能运行浏览器，并支持JavaScript的浏览器，就可正确执行，从而实现了“编写一次，走遍天下”的梦想。

本项目的前端交互应用了javascript语言，提高良好的交互性能，减少了后台数据的压力，对表单的判断等都可由前端直接执行判断。在调试的时候直接使用浏览器进行调试，方便开发。

### 2.2.2 jquery

JQuery是一个javascript的框架，是对javascript的一种封装。使用jquery可以消除浏览器差异，不需要写冗长的代码来针对不同的浏览器来绑定事件，编写AJAX等代码，可以简洁的操作DOM的方法，轻松实现动画、修改CSS等各种操作。

本项目前端使用了jquery来实现遍历HTML文档，操作DOM、事件处理、执行动画等操作，相比javascript更加简洁，更加方便。

### 2.2.3 百度地图开发

百度地图JavaScript API是一套由JavaScript语言编写的应用程序接口，可在网站中构建功能丰富、交互性强的地图应用，支持PC端和移动端基于浏览器的地图应用开发，且支持HTML5特性的地图开发，支持HTTP和HTTPS，免费对外开放，可直接使用。在申请百度AK之后，接口无限制使用。

本项目开发的应用百度地图进行定位，提供路径导航，获取用户的地理位置信息，并为其展示周边的停车场。相比腾讯地图，百度地图开发文档更加完善，且有丰富的demo，能很快的进行开发应用。

## 2.3微信公众号开发

成为开发者消息校验原理

首次提交验证申请时，微信服务器发送GET请求到填写的URL上，并且带上四个参数（signature、timestamp、nonce、echostr），开发者通过对签名（即signature）的效验，来判断此条消息的真实性。此后，每次开发者接收用户消息的时候，微信也会带上前面三个参数（signature、timestamp、nonce）访问开发者设置的URL，开发者依然通过对签名的效验判断此条消息的真实性。效验方式与首次提交验证申请一致。

开发者通过检验signature对请求进行校验，若确认此次GET请求来自微信服务器，请原样返回echostr参数内容，接入生效，成为开发者成功，否则接入失败。

## 2.4 本章小结

本章主要介绍微信公众号开发及其需要的技术。首先介绍了微信公众号平台基础知识，结合项目的开发，ssh（spring，springMVC,hibernate）框架涉及到的概念，包括数据交换格式，前端开发用到的主要技术。

# 第三章 应用需求分析

根据上文项目背景与应用分析的简要阐述，设计项目的基本需求，本章将会通过需求分析来进一步确定应用的主要功能及设计目标。

## 3.1系统应用需求分析

根据项目的要求，结合公众号的开发，本应用需要提供以下功能：

（1）用户注册登录：用户在关注公众号之后，需要完善自己的信息，进行注册登录；

（2）根据用户位置定位，地图上显示停车场：根据当前所在位置，显示最近的20个停车场；

（3）停车场搜索：用户根据需要，输入位置，根据位置显示最近的20个停车场；

（4）停车场管理：用户发布停车场，管理停车场，进行修改，删除；

（5）管理车辆：用户需要添加车辆，对自己的车辆进行管理。

（6）预定停车场：用户根据需要，可以对停车场进行预定，取消预定。

根据用户的需求，相应的用例图如下3-1。

（7）管理员模块：对用户信息进行审核，验证。

（8）更改公众号菜单：对公众号的界面菜单就行更改，删除。

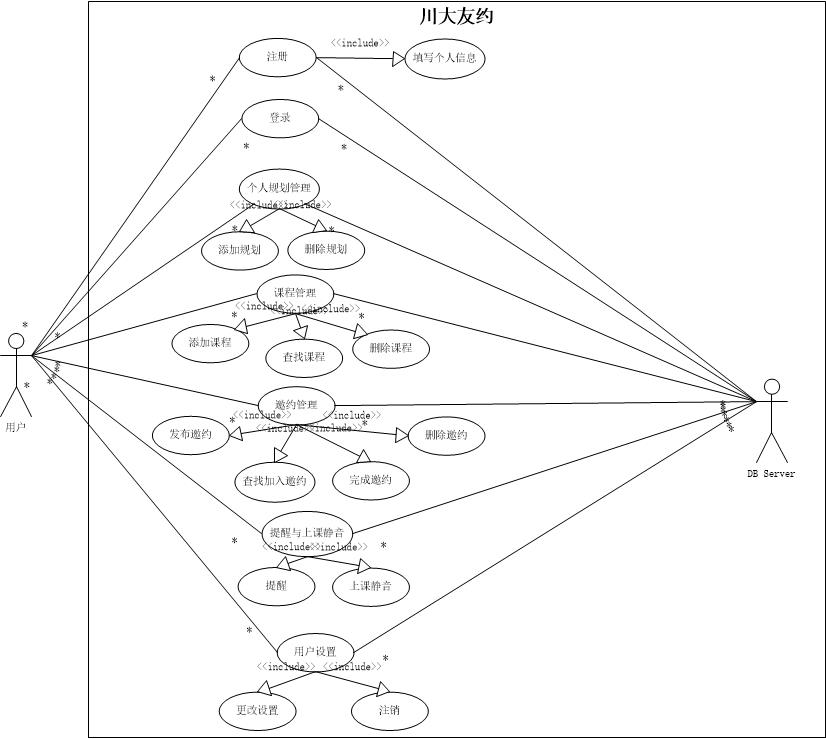


图3-1应用整体用例图

## 3.2性能需求

系统的性能在开发设计过程中是一个非常重要的过程，好的系统不仅可以提升系统的质量，还可以提升用户的认可程度，尤其在数据库的设计与开发，应尽可能减少对数据库的操作，得到尽可能多得数据，减少系统压力，以防因访问量过多，系统崩溃。根据系统设计的目标，本公众号的性能需求如下：

1. 系统性能需求

本应用是公众号，运行在微信上边，在公众号上访问服务器反应时间要求较高，尤其应能及时准确的加载定位，实时传递应用数据，不能要用户等待过长，如果有等待不能影响其他用户的使用。该系统需要多次请求获取数据，对数据的交互以及准确度要求非常高。

1. 运行性能需求

该应用主要运行在移动设备上边，由于设备功能资源限制，应减少运行时所占内存，降低所占用的cpu资源。

1. 界面功能需求

本应用主要针对广大的车主，界面要求尽可能的布局合理，简洁美观，方便用户操作。

1. 安全性需求

系统需确保用户数据安全，用户停车记录、个人信息等保密。

1. 实时性能需求

实时性能至关重要,在软件的需求分析中充分考虑,以保证开发出的应用软件的实时性能。这些实时性能的体现一部分来源于实时操作系统的实时性能,另一部分则依赖于应用软件本身的设计和代码。本应用实时性能要求较高，需设计满足要求的系统。

## 3.3 公众号设计目标

根据上述用例和需求，该公众号的具体设计目标为：

（1）实时加载用户位置，停车场数据

（2）用户能及时进行停车场预约、取消，查看自己的订单

（3）根据用户的位置，提供到停车场的导航

（4）用户能方便添加、查看、删除、修改停车场

（5）界面简洁，易于操作

开发实现中上述目标的优先级别依次降低。

## 3.4 本章小结

本章明确了应用的功能性需求，同时描述了性能需求并确定公众号设计目标。

# 第四章 应用概要设计

“微信智能停车场”是基于微信开发的公众号，主要是针对广大车主及有停车位的用户开发的应用，旨在为广大车主提供停车场停车，为广大有停车位的用户出租停车场。该应用主要解决车车主找停车位难，有停车位的用户难出租的问题。本章将根据需求分析进行系统模块与功能设计，以及数据库设计的详细说明。

## 4.1 应用功能模块

根据用户需求， 本项目所包含的功能模块如下图4-1所示：



图4-1应用功能模块

整个项目主要分为两个端系统，客户端负责面向用户进行交互和简单的处理操作，服务器端负责数据处理与存储。由于客户端集合了所有面向用户的功能，因而此处着重介绍客户端的功能模块设计，服务器端将于4.3中予以介绍。

客户端分为了四大模块，分别为用户模块、网络模块、数据库模块、爬虫模块。其中，面向用户的主要模块为用户模块，实现了整个应用的主要功能；网络模块是客户端与服务器端通信的接口模块，负责与服务器建立连接、向服务器传输用户数据并接收服务器响应的用户数据等；数据库模块是为避免客户端频繁与服务器端进行数据交互而降低性能，因而将服务器端传回的用户数据在本地进行备份，使得用户可以离线加载数据，在本地也拥有缓存数据；爬虫模块保证了应用能够根据用户教务处账号密码找寻、爬取并解析其课表数据。

以下介绍用户模块，该模块实现了本应用所有面向用户的功能，其子模块如下图4-2。

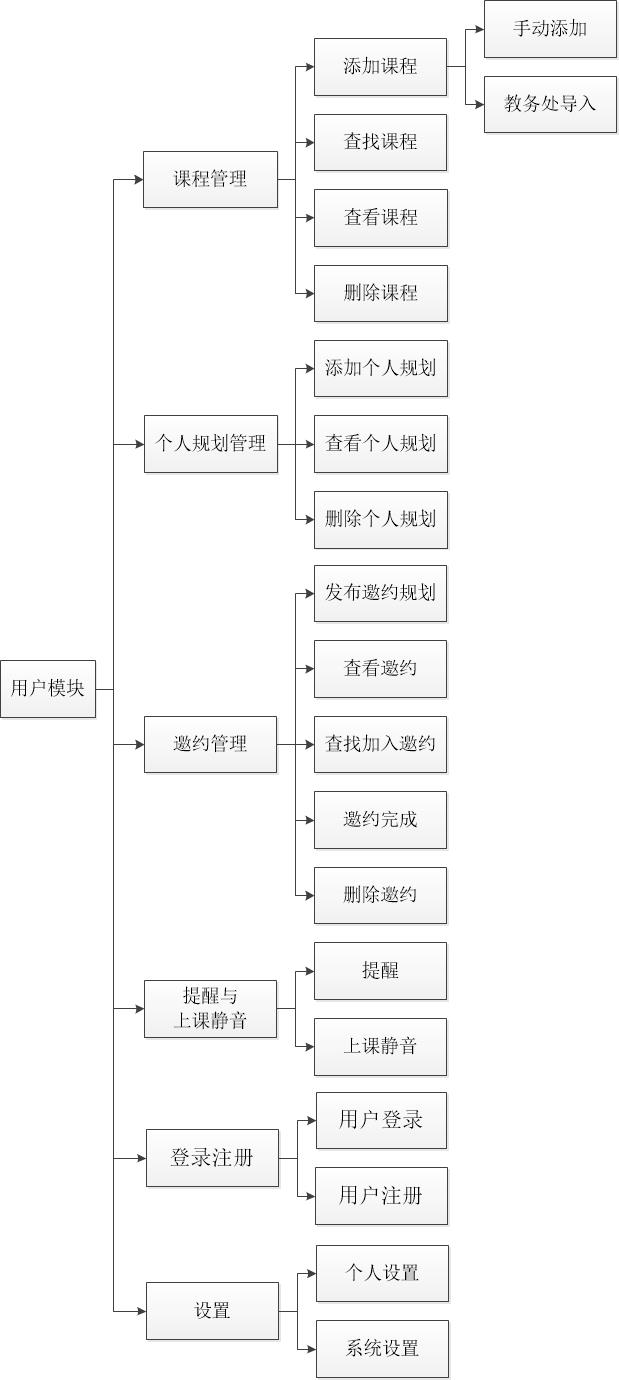


图4-2用户模块功能

用户模块包含了本应用的所有用户操作，分为了六大部分，其中包括登录注册模块、课程管理模块、规划管理模块、邀约模块、提醒与上课静音模块与设置模块。

### 4.1.1 用户关注公众号模块

用户关注公众号，获取公众号详细，其主要操作如下：

扫二维码关注：用户扫描二维码之后，获取恢复详细，关注公众号成功，用户可以进行对公众号的操作。

### 4.1.2自定义公众号菜单模块

个人规划管理分为添加规划、查看规划与删除规划。此处个人规划仅自己可见，主要方便了用户可以根据个人自身需要量身定制自己的规划表。主要操作如下：

1）添加个人规划：用于将个人规划添加至课余时间中，需要输入：规划类型、规划名称、规划日期、规划时间、规划地点、规划详细内容。其中规划类型分为：晨读规划、学习规划、健身规划以及自定义规划。规划日期会根据当前开学日期自动转换为教学周数，方便用户查看。添加规划提交时，若规划时间与上课时间冲突，则会给出提醒，提示用户添加失败请重新修改时间。

2）查看个人规划：在主界面可以查看到当前用户的个人规划情况，采用紫色背景。点击其中的规划格子就可以查看到当前的规划详细信息，即可查看到用户添加时输入的所有信息。

3）删除个人规划：在个人规划详细信息中可以选择删除个人规划，则从该用户的规划表中删除掉。

### 4.1.3 邀约管理模块

邀约模块是为了用户能够更高效地完成规划任务而设置的一个重要功能。在主界面上，邀约规划有四种颜色以供用户区别查看，蓝色代表该邀约规划正在结伴邀请中，此时该邀约会出现在邀约查找结果中，允许用户查找添加，添加过该邀约的用户可以在邀约的详情界面中查看到此时的参与人数；红色该邀约规划正在执行中，此时用户无法继续添加该邀约了，即无法从邀约查找结果中查找到；灰色代表该邀约规划时间已结束，但用户在详情界面未选择或选择未完成该规划任务；浅绿色代表该邀约规划时间已结束，用户选择已完成该邀约的任务规划。

邀约模块主要包括发布邀约规划、查找并加入邀约、查看已发布的邀约、完成邀约规划与删除邀约功能，主要操作如下：

1）发布邀约规划：用户将自己添加的未来时间的规划任务发布到邀约平台上，其他用户可以在搜索中找到该邀约规划。发布成功的邀约规划会自动添加到自己的邀约规划中，并在主界面上呈现蓝色背景，表示正在邀请中。

2）查找并加入邀约：用户加入别人邀约的唯一渠道。用户可以在查找邀约界面中查看到当前所有未开始的邀约规划，也可以选择邀约的类型、名称、周数、日期、时间、地点按条件查找，相关邀约搜索结果会以列表形式列出。用户可以在搜索结果中选择邀约加入，此时其他用户发布的邀约就会出现在该用户的主界面中。

3）查看已加入的邀约：在主界面点击邀约格子就可以查看邀约规划的详细信息，邀约的详细信息包括：邀约类型、邀约名称、邀约日期、邀约时间、邀约地点、邀约详细信息、邀约参与人数、规划完成情况（未开始、进行中、未完成、已完成）。邀约完成后，还会出现规划完成情况选项框，可以选择完成或未完成；同时会出现点赞按钮，可以为此次邀约点赞，并可以看到为此次邀约点赞的用户的用户名。

4）邀约完成：当系统时间达到邀约规划结束时间之后时，邀约自动结束，在主界面呈现灰色背景，用户可以在邀约详细界面选择自己是否已完成此次邀约规划，当选择已完成邀约规划后，该邀约规划在主界面的背景会变成绿色。同时，可以在邀约详细界面为此次邀约点赞，也可以看到同样为该邀约点赞的用户的用户名。

5）删除邀约：用户删除邀约后，只会影响自己的邀约规划表，对于其他也参与了此次邀约的同学而言并不受影响，在邀约详细界面会看到人数减一。

### 4.1.4 提醒与上课静音模块

提醒是本应用为了帮助用户方便查看即将要开始的课程或规划的信息而设置的重要功能，当在用户设置的课前或规划前时间时，便会以通知栏提醒的方式，提醒用户注意课程或规划开始时间与地点，以免错过课程与规划。

上课静音模块是在上课期间自动将用户手机调至静音状态，以免上课期间电话或短信声音打扰到自己、老师与其他同学，并在下课时间将用户手机自动恢复正常状态。

### 4.1.5 登录注册模块

登录注册模块分为用户注册、用户登录，用户注册可以为该应用创建一个属于自己的新账号，用于记录之后的用户课程信息、规划信息以及邀约信息等。

### 4.1.6 设置模块

设置模块实现了用户对本应用的所有设置，其中系统设置包括对提醒功能及上课手机静音功能的设置，用户可以选择是否要开启，还可以进行提醒时间的查看与更改等设置。个人设置包括对用户登录密码的更改、用户注销等操作。

## 4.2 客户端总体架构

Android应用开发中鼓励弱耦合和组件的重用，逻辑与视图的分离是Android程序设计的基本原则。因此，本应用客户端采用传统的MVC（model-view-controller，模型-视图-控制器）框架模式，解耦隔离界面显示与业务逻辑和数据，实现了代码的高内聚低耦合。其主要分为三层：视图层（View），控制器层（Controller），模型层（Model）。[[[1]](#endnote-2)]以下是本应用的框架结构图：

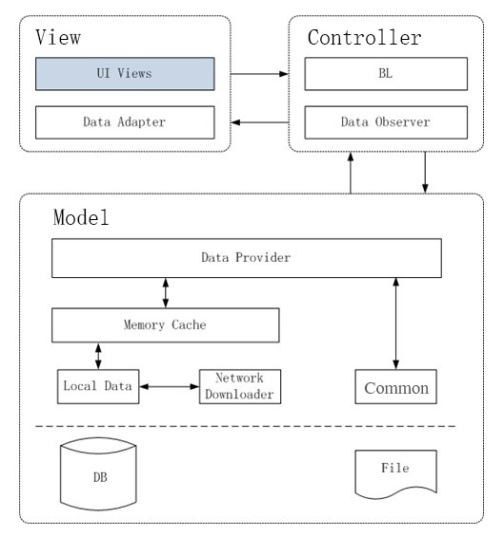


图4-3 Android应用框架图

1）视图层（View）

视图层主要负责数据展示并提供与用户进行交互的界面，提供用户输入的接收与结果的显示。Android中为了减少耦合度，将逻辑与视图相分离，因此一种通用的做法是在xml布局文件中编写界面视图，再通过Activity引入。

本应用的布局文件全部放置在res\layout中，通过xml来进行界面的布局编写，并在布局中添加相应控件，设置属性以达到布局的目的。View 类是所有视图控件 （包括 ViewGroup） 的基类。各种 UI 控件都继承 View类，通过实现不同的接口实现特定功能。[[[2]](#endnote-3)]本应用为了实现界面设计的三大原则：置界面于用户的控制之下、减少用户的记忆负担、保持界面的一致性，并达到统一的设计风格，使用了Material Design风格来进行设计，并封装了许多自定义控件，使得界面更加美观。

2）控制层（Controller）

控制层主要负责处理与用户交互的控制逻辑以及模型层的处理事件。本应用中，控制层主要由Activity来实现，其中主要监听视图层，根据用户请求响应触发事件并控制用户界面数据显示，同时将状态改变告知模型层。

3）模型层（Model）

模型层是应用的主体部分，负责处理应用的业务逻辑，如：和数据库的交互、数据处理、网络操作及部分耗时任务等。

本应用中客户端功能模块中的网络模块、爬虫模块与数据库模块都是在此层中实现的。客户端的网络模块通过OkHttp框架进行安全的http请求的发送与服务器端响应的接收；客户端数据库模块是通过Android所带的sqlite数据库来缓存一部分用户数据，同时也使用了Android中的SharePreferences接口来实现部分设置信息的存储；客户端爬虫模块通过jsoup库封装Http请求进行四川大学综合教务系统的登录，之后解析并提取HTML文本内容，实现了应用爬取用户数据的功能。

## 4.3 服务器端架构与数据库设计

根据应用功能模块的划分，服务器端主要承担了用户数据处理以及存储的功能，包含了网络模块与数据库模块。为了更好的专注于逻辑处理功能，本项目服务器端采用了Struts2的架构进行设计。

### 4.3.1 框架与响应流程

Struts 2框架由3个部分组成：过滤器（FilterDispatcher）、业务控制器、和用户实现的业务逻辑组件。过滤器负责拦截所有的用户请求，当请求是\*.action结尾会被转入Struts2框架处理，Struts2再决定调用哪个业务逻辑组件。业务控制器就是实现Action类的实例，Action类通常包含一个execute方法（也可在配置文件中指定方法执行），该方法返回一个逻辑视图名的字符串。

在Struts 2框架中，实际上是包含了真正意义上的Struts2和Xwork两种框架，Struts2的职责是对Http请求进行处理后，委托XWork完成真正的逻辑处理。 [9]其响应流程如下：

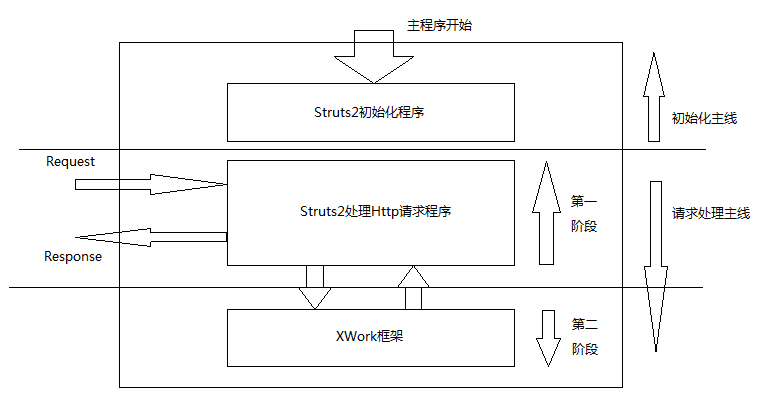


图4-4 Struts2框架响应流程

其中，过滤器部分属于Struts 2 中Http请求的处理部分，而业务逻辑则是XWork的职责，本项目的服务器端就是通过实现action来完成用户的业务逻辑处理，主要为与服务器端数据库的连接并对数据库进行CRUD操作。

### 4.3.2 数据库设计

本项目中数据是应用的基础，本应用的用户信息、个人信息、规划信息、课表信息及邀约信息等数据都将存储在服务器端的Mysql数据库中，并同时在本地客户端进行缓存备份。因此服务器端的数据库设计是整个项目的重中之重，以下图4-7是服务器端的数据存储E—R图：

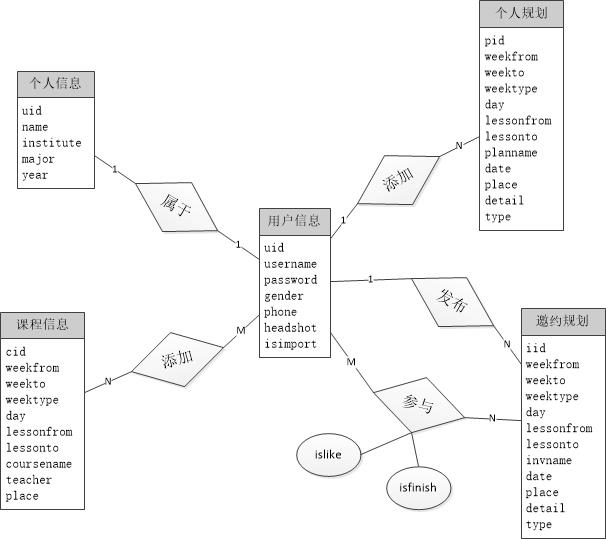


图4-5 E-R图

根据E-R图的概念模型转换为符合Mysql所支持数据模型的逻辑结构，形成六张数据表，分别为表4-1用户信息表user\_msg、表4-2个人信息表personal\_info、表4-3规划信息表plan\_info、表4-4邀约信息表inv\_info、表4-5用户邀约表user\_inv、表4-6课程信息表course\_info与表4-6用户课程表user\_course。

表4-1 用户信息表 user\_msg

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 意义 |
| uid | Integer | 主码，自增，不为空 | 用户序号 |
| username | varchar(20) | 不为空 | 用户名 |
| password | varchar(20) | 不为空 | 密码 |
| gender | varchar(20) |  | 性别 |
| phone | varchar(20) |  | 电话 |
| headshot | varchar(20) |  | 头像 |
| isimport | Integer | 不为空 | 是否绑定过教务处 |

表4-2个人信息表personal\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 意义 |
| uid | Integer | 主码，外码（参照user\_msg），不为空 | 用户序号 |
| name | varchar(20) |  | 真实姓名 |
| institute | varchar(20) |  | 学院 |
| major | varchar(20) |  | 专业 |
| year | varchar(20) |  | 入学年级 |

表4-3规划信息表plan\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 意义 |
| pid | Integer | 主码，自增，不为空 | 规划信息序号 |
| weekfrom | Integer |  | 规划起始周数 |
| weekto | Integer |  | 规划结束周数 |
| weektype | Integer |  | 周数类型 |
| day | Integer |  | 星期几开始规划 |
| lessonfrom | Integer |  | 规划开始节次 |
| lessonto | Integer |  | 规划结束节次 |
| planname | varchar(20) |  | 规划名称 |
| pplace | varchar(20) |  | 规划地点 |
| pdetail | varchar(20) |  | 规划详细内容 |
| ptype | Integer | 不允许为空 | 规划类型 |
| puid | Integer | 主码，外码（参照user\_msg），不为空 | 发布该规划的用户序号 |

表4-4 邀约信息表inv\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 意义 |
| iid | Integer | 主码，自增，不为空 | 邀约信息序号 |
| weekfrom | Integer |  | 邀约起始周数 |
| weekto | Integer |  | 邀约结束周数 |
| weektype | Integer |  | 周数类型 |
| day | Integer |  | 星期几开始邀约 |
| lessonfrom | Integer |  | 邀约开始节次 |
| lessonto | Integer |  | 邀约结束节次 |
| invname | varchar(20) |  | 邀约名称 |
| place | varchar(20) |  | 邀约地点 |
| detail | varchar(20) |  | 邀约详细内容 |
| type | Integer | 不允许为空 | 邀约类型 |
| isfinish | Integer | 不允许为空 | 邀约是否已过结束时间 |

表4-5 用户邀约表user\_inv

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 意义 |
| uid | Integer | 主码，外码（参照user\_msg），不为空 | 用户序号 |
| iid | Integer | 主码，外码（参照inv\_info），不为空 | 邀约序号 |
| isfinish | Integer |  | 邀约规划完成情况 |
| islike | Integer |  | 是否点赞该邀约规划 |

表4-6课程信息表course\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 意义 |
| cid | Integer | 主码，自增，不为空 | 课程信息序号 |
| weekfrom | Integer |  | 从第几周 |
| weekto | Integer |  | 到第几周 |
| weektype | Integer |  | 单/双周上课 |
| day | Integer |  | 星期几上课 |
| lessonfrom | Integer |  | 从第几节 |

（续表4-6）

|  |  |  |  |
| --- | --- | --- | --- |
| lessonto | Integer |  | 上到第几节 |
| coursename | varchar(20) | 不为空 | 课程名称 |
| teacher | varchar(20) |  | 授课教师 |
| place | varchar(20) |  | 上课地点 |

表4-7用户选课表user\_course

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 意义 |
| uid | Integer | 主码，外码（参照user\_msg），不为空 | 用户序号 |
| cid | Integer | 主码，外码（参照course\_info），不为空 | 课程信息序号 |

客户端SQLite数据库字段与服务器端相同，但是仅存储从服务器端传回的部分数据，作为本地缓存使用。

## 4.4本章小结

本章主要根据应用的需求分析，将应用的功能模块进行了划分，并具体阐释了各功能模块的概念设计，并设计了客户端的总体架构，同时给出了服务器端的概要设计，包括框架与响应流程，以及数据库设计。

# 第五章 应用详细设计和实现

“川大友约”应用的客户端是通过Android Studio开发平台用Java语言基于Android 7.0 N api 24实现的，服务器端是通过Eclipse平台基于Java EE开发实现的。下面将从客户端和服务器端分别来介绍整个应用的详细设计。

## 5.1 客户端的详细设计与实现

根据概要设计阶段所划分的功能模块，采取MVC架构进行设计并基于Android 7.0 （2016年5月正式发布）使用api 24开发，具体实现的代码目录结构如下图：

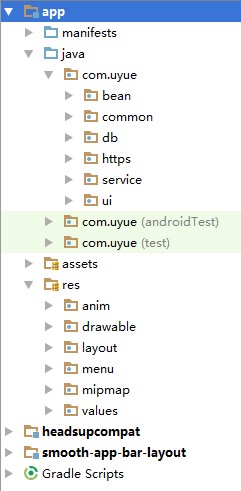


图5-1 代码目录结构

其中，res目录中放置了项目中所用到的所有资源，包括图片、布局、字符串等资源。

java目录中放置所有以java代码实现的主程序以及android测试代码，主程序中包ui实现了界面层与业务逻辑层，其中包含：

1）Activity：负责将引入xml文件所描述的页面布局并将UI视图加载以及设置监听，同时将业务逻辑交由Model层进行处理。

2）Adapter：定义多重复合布局，可以根据后台数据而决定在前端显示的Item，同时可以优化绘制界面，提高性能。

3）Widget：将自定义控件统一化规范化，提高了代码的重用性，实现了应用界面设计的一致性。

包db、https和service实现了模型层（Model），其中包含：

1）dao：封装了对每张数据表的CRUD操作，负责与本地数据库进行数据存取，。

2）https：其中HttpUtil类，封装了okhttp3的sendOkHttpRequest方法，使其能够在子线程中直接调用来访问服务器。NetHelper类实现了爬取教务处网站的个人信息和课程信息，以及对爬取下来的元数据进行jsoup解析，并封装成对象返回。

3）service：实现程序后台运行服务的方案，实现了上课时间提醒与各种数据变化的通知机制。其中有ReminderListenerService类与SilentListenerService类，二者都继承自Service类，用于创建后台进程并实现时间的监听，同时在service中启动广播接收器，使得广播接收器可以在程序退出后在后台继续执行，来接收系统时间变更的广播事件。

包common完成了辅助工具类的实现，包括一些复杂计算以及管理工具。

以下将按照应用功能模块详细介绍应用的设计与实现。

### 5.1.1 滑动菜单的设计与实现

滑动菜单是Material Design中推荐的做法，所谓滑动菜单就是将一些菜单选项隐藏起来，而非放置在主屏幕上，可以通过滑动的方式将其唤出。这种方式既节省了屏幕空间，能够让用户专注于核心功能的操作，又实现了非常好的动画效果，提升了用户体验。

本应用的滑动菜单栏隐藏在主界面的左侧，用户通过点击左上角的菜单按钮可以显示菜单栏，同时支持用户在屏幕左侧边缘向右拖动来显示菜单栏。当用户向左滑动菜单栏或点击菜单栏以外的区域后，菜单栏会自动隐藏，从而回到主界面。滑动菜单的界面原型如下图：



图5-2 菜单栏界面原型

显示滑动菜单界面布局使用了由support-v4提供的DrawerLayout，在其中放置了两个直接子控件，其中FrameLayout用于显示主界面的界面布局，而NavigationView用于在滑动菜单页面显示自定义布局。自定义布局中包含菜单标头和菜单选项，菜单标头由nav\_header.xml实现，其中显示了当前日期及用户名、用户所在学院和用户头像；菜单选项由nav\_menu.xml实现，其中显示了对规划的管理操作以及设置等选项。

界面布局后在主界面引入并获取到相应的实例，再设置左上角菜单按钮点击事件的监听器，若有点击则调用DrawerLayout的openDrawer()方法将滑动菜单显示出来，之后调用setNavigationItemSelectedListener()方法来监听菜单的选中事件，当用户点击了任意菜单项时，就会回调到onNavigationItemSelected()方法中实现相应的逻辑处理。

### 5.1.2 动态显示主界面的设计与实现

为了以格子方式显示主界面，首先要通过initTable()方法初始化格子布局文件，再通过getInfoFromServer()方法从服务器端取回该用户的课程、规划及邀约信息并初始化列表。由于用户的课程、规划、邀约信息都是以Map<int,ArrayList>形式保存在变量中的，通过initTableBody()方法将每天的信息取出，剔除不在该周的信息后，进行开始节次的排序，再进行时间冲突的检测，计算信息的放置位置并以TextView进行信息的动态显示，重叠信息则自定义gallery形式显示。

这样就可以根据课程周数、星期与时间以及规划的日期与时间依次排序显示在格子中。在主界面中通过setBackgroundResource()方法设置不同的背景色，课表的背景颜色使用橙色，个人规划背景色使用紫色，而邀约的背景色设置则需调用当前系统时间与邀约时间作对比后再显示。

### 5.1.3 爬虫的设计与实现

本应用客户端为了实现教务处一键导入课程，使用了可以爬取用户教务处系统数据的爬虫模块，该模块嵌入在课程管理模块中使用。此模块是由NetHelper类来实现，利用jsoup的connect()方法连接教务处，将用户的用户名与密码用post方法上传后按标签解析HTML页面，若登录成功则返回JSESSIONID的值，若登录失败则返回错误代码提示用户重新登录。由于学分制综合教务系统有登录session的限制，因此，为了用户的账号密码安全，本应用仅将其保存着本地数据库中，作session过期后的刷新重登录之用，无需用户再次输入账号密码，提高用户体验。在NetHelper类的实例中实现了从教务处网站爬取用户课程信息的功能，直接利用带有session参数的cookie连接教务处的学生课表页面，解析返回的HTML页面获取该用户所选的全部课程并返回对象。

### 5.1.4 课程管理模块

1）添加课程分为从教务处导入和手动添加课程。

教务处导入课程：在用户请求从教务处导入课程信息时，就需要得到用户的许可权限，因此，在LoginJwcActivity类中，当用户输入了入教务处账号以及密码后，开启登录子线程。交由后台封装的爬虫模块实现，最终得到了爬取数据的对象并将其上传至服务器，再存入本地数据库并显示于主界面中。爬虫模块的具体实现于5.1.3中已经介绍。

手动添加课程：由CourseAddActivity类实现，用户输入课程名、任课教师后，通过滑动滚轮选择上课周数、上课时间以及上课地点。由isConflict()方法检测规划是否与现有规划或邀约冲突，若无则通过子线程将用户添加的合法课程上传至服务器后，再插入本地数据库中并显示于主界面中。

2）查找课程：CourseSearchActivity类在初始化ui时就开启子线程发送HTTP请求访问服务器，从服务器查找当前用户的所有未选择课程，以JSONArray形式传回客户端，客户端将JSON字符串解析后设置CourseSearchAdapter的适配器并以RecyclerView的形式显示出来。

同时，查找课程还采用了关键词检索方式，用户可以输入与课程相关的任何关键词进行模糊检索，客户端将用户输入的关键字传给服务器端在数据库的course\_info表中进行搜索，将搜索结果传回，形式同上。

用户可以在查找出的课程中选择加入课程，客户端获取该课程的cid后上传服务器并插入本地数据库，后跳转至主界面显示加入后的课表。

3）查看课程与删除课程：该子模块集成在查看课程详细信息CourseDetailInfoActivity中，用户在主界面上点击某课程后，CourseActivity通过Intent将课程对象以Serializable的形式传给该活动，该活动从Intent中取回课程内容显示在界面上。当用户点击删除课程按钮时，将课程cid上传至服务器并删除数据库中user\_course表的相关记录，将删除结果传回并刷新主界面显示。

### 5.1.5 个人规划管理模块

1）添加个人规划：由PlanAddActivity类实现，用户选择或输入个人规划类别后，输入规划名，再通过滑动滚轮选择规划日期、规划时间以及规划地点，最后输入规划的详细情况点击“个人规划”按钮。通过Utility中getWeeksByDates()方法将规划日期按开学日期转换为教学周后，由isConflict()方法检测规划是否与现有课程或邀约冲突，若无则将用户添加的个人规划上传至服务器后，再插入本地数据库中并显示于主界面中。

2）查看与删除个人规划：该子模块集成在查看个人规划详细信息PlanDetailInfoActivity中，用户在主界面上点击某规划后，CourseActivity通过Intent将规划对象以Serializable的形式传给该活动，该活动从Intent中取回规划内容显示在界面上。当用户点击删除个人规划按钮时，将规划pid上传至服务器并删除数据库中plan\_info表与user\_plan表中的相关记录，将删除结果传回并刷新主界面显示。

### 5.1.6 邀约模块

1）发布邀约：由InvAddActivity类实现，用户选择或输入邀约规划类别后，输入邀约名，再通过滑动滚轮选择邀约日期、邀约时间以及邀约地点，最后输入邀约的详细情况点击右上角按钮。通过Utility中getWeeksByDates()方法将规划日期按开学日期转换为教学周后，由isConflict()方法检测规划是否与现有课程或个人规划冲突，若无则将用户添加的邀约规划上传至服务器后，再插入本地数据库中并显示于主界面中。

2）查找并加入邀约：由InvSearchActivity类实现，通过子线程访问服务器来查找所有已发布的邀约，将查询的邀约信息以jsonArray形式传回客户端并显示。

若邀约信息太多，用户也可选择条件查找，由InvCriteriaSearchActivity类实现，用户可以输入要查询的规划类型，选择规划周数、时间或地点其中的任意几项进行查询，点击“查询”后，客户端将用户输入的关键字传给服务器端，服务器端在数据库的inv\_info表中进行搜索，将搜索结果传回，形式同上课程查找。

用户可以在查找出的邀约中选择加入邀约，客户端获取该邀约规划的iid后上传服务器并将邀约规划信息插入本地数据库，后跳转至主界面。同时，服务器也传回该邀约发布人的uid，当该用户与服务器再次取得联系后，服务器端将向该uid的用户的客户端发送邀约成功信息，客户端显示该消息通知。

3）查看已加入的邀约：在主界面点击邀约格子转到InvDetailInfoActivity界面，主界面活动CourseActivity通过Intent将邀约规划对象invInfo以Serializable的形式传给InvDetailInfoActivity，该活动从Intent中通过getExtra()方法取回邀约规划内容显示在界面上。

4）完成邀约：用户在InvDetailInfoActivity界面中时，该活动会通过isTimeToEnd()函数取得当前系统时间与该邀约规划的日期时间做对比，若当前时间超过了邀约时间则邀约已结束，界面中的完成按钮Enable置为true，用户可以点击按钮选择完成与否，选择完成则按钮Enable置为false，并将iid、uid与赋为1的isfinish值通过子线程上传至服务器，返回上传结果后，主界面将该邀约灰色背景换为绿色；若选择未完成则按钮Enable置为false即可。同时邀约结束后，InvDetailInfoActivity会有ShowPeopleLike()方法显示为当前邀约点赞的用户的用户名，通过将iid上传服务器在user\_inv表中查询islike字段为1的用户并取回其user\_name显示在界面中。

5）删除邀约：当在查看邀约界面点击删除邀约按钮时，会触发deleteInv()方法，客户端将iid和uid上传至服务器并查找数据表user\_inv并将其中符合记录删除，将修改结果传回并刷新主界面显示。

### 5.1.7 提醒与上课静音模块

提醒模块由ReminderListenerService类实现，上课静音模块由SilentListenerService类实现，二者都继承自Service类，用于创建后台进程并实现时间的监听，由onCreate()方法创建后台进程，由onStartCommand()通过IntentFilter的Intent.ACTION\_TIME\_TICK参数注册并启动一个广播接收器BroadcastReceiver，来实现时间变更事件的监听，其变更间隔为1min，恰好在该应用的时间误差范围内。

提醒模块中通过isReminderTime()方法通过计算用户课程列表与规划列表中的每一时段的开始时间与设置的提醒时间之差，再和当前系统时间做对比，得出是否要为用户发送消息栏通知；getNotification()方法实现了为当前用户发送消息栏通知的功能。

静音模块中isSilentTime()方法通过计算用户课程列表中的每一节课的上课时间与当前系统时间比较后得出是否要为用户设置静音模式或为用户取消静音模式；silentSwitchOn()方法实现了将手机设置静音的功能，silentSwitchOn()方法实现了将手机恢复正常响铃状态的功能。

### 5.1.8 登录注册模块

本模块主要实现了个人账户管理的功能，用户可进行用户注册、登录以及个人设置等操作。

1）用户注册：未注册用户在AppRegisterActivity实现的注册界面中，输入用户名、输入两次密码、手机号并选择性别后点击注册按钮，客户端验证用户输入合法后将信息上传至服务器端存储在user\_msg表中，之后返回注册结果。

2）用户登录：已注册用户可以直接在AppLoginActivity实现的登录界面中输入账号密码登录，客户端将登录信息上传至服务器端验证用户信息后返回登录结果，当登录成功后本地缓存用户名与密码方便用户二次免密登录。

### 5.1.9 设置模块

设置模块SettingActivity类实现了用户对本应用的所有设置。

1）个人设置：可以进行密码修改以及退出登录的操作。密码修改passwordChange()方法接收所输入的原密码以及输入的新密码后，经服务器端验证原密码正确后进行数据库对密码字段的更新。退出登录userLogout()方法实现了清空本地缓存的用户信息并跳转应用登录界面，使用户可以重新登录。

2）提醒与上课静音的设置：在API23以上后，android加强了消息安全机制，需要声明ACCESS\_NOTIFICATION\_POLICY权限来触发Do Not Disturb界面，来向用户申请访问手机消息设置的权限，getDoNotDisturb()方法是获取Do not disturb权限,才可进行手机静音操作；initListener()方法对该界面所有的按钮注册监听器进行点击事件的监听，实现了对提醒开关和上课静音开关以及提醒时间设置的监听；setSilentOn()启动后台静音服务，监听系统时间变化；setSilentOff()关闭后台静音服务并取消广播监听器注册；setReminderOn()启动后台提醒服务，监听系统时间变化；setSilentOff()关闭后台提醒服务并取消广播监听器注册。

## 5.2 服务器端的详细设计

本应用的服务器端使用Struts2框架，运用分层开发技术，结构清晰。服务器端主要负责将客户端所传上来的数据进行分析处理，并访问服务器端数据库进行CRUD操作，最后将处理结果传回客户端。其代码结构如下图：

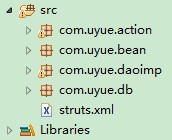


图5-3 服务器端代码结构

在Struts2框架中，struts.xml文件起到FilterDispatcher的作用，负责拦截所有发送到服务器的资源请求（即Android客户端发送的请求），当请求时\*.action时会被转入Struts2框架进行处理，分流到每个action后所请求的操作会被执行，最后将结果封装成json格式返回给用户。

本服务器端的主要业务逻辑可以分为两层：控制层与数据处理层。控制层主要实现了与客户端交互的接口，其中包括action包；处理层主要提供数据处理操作，其中包括daoimp包，db包。下面根据业务逻辑层级来介绍服务器端的设计与实现方式。

### 5.2.1 控制层

控制层中最重要的部分是action类，负责过滤后的用户请求的传入与处理。以下是控制层的主要类图：

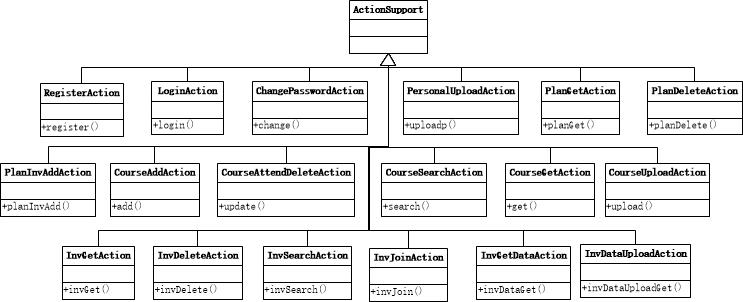


图5-4 控制层主要类图

其中，主要类的主要功能如下：

1）RegisterAction写入新用户的登录信息；

2）LoginAction验证用户的登录信息；

3）PersonalUploadAction 将用户从教务网站的个人信息写入personal\_info数据表中；

4）ChangePasswordAction接收用户的原密码进行比对并修改成新密码

5）PlanGetAction获取当前用户的所有规划信息并返回；

6）PlanInvAddAction将用户加入的个人规划或邀约写入规划信息表plan\_info中或邀约表inv\_info中与user\_inv中；

7）PlanDeleteAction将用户要删除的规划从plan\_info中删除；

8）CourseAddAction将用户添加的课程信息写入课程信息表course\_info与用户课程表user\_course；

9）CourseAttendDeleteAction将用户加入的课程写入用户课程表user\_course以及将用户要删除的课程从用户课程表user\_course中移除；

10）CourseSearchAction查找用户未选的所有课程或查找关键字课程并返回；

11）CourseGetAction将该uid用户所选的所有课程列表全部返回；

12）CourseUploadAction将用户从教务处网站上爬取的用户所选课信息写入数据库中；

13）InvGetAction将该uid用户所加入的所有邀约列表全部返回；

14）InvDeleteAction将用户要删除的邀约从用户邀约表user\_inv中移除；

15）InvSearchAction将用户的无条件查找或有条件查找的邀约结果信息返回；

16）InvJoinAction将用户的选择加入的邀约添加到user\_inv中；

17）InvDataGetAction从服务器取回该邀约信息的详细情况，包括参与人数、完成情况与点赞人数等。

18）InvDataUploadAction将该uid用户的完成情况与点赞情况更新到用户邀约表user\_inv中。

### 5.2.2 数据处理层

数据处理层负责与数据库的连接访问，进行数据处理与数据操作。

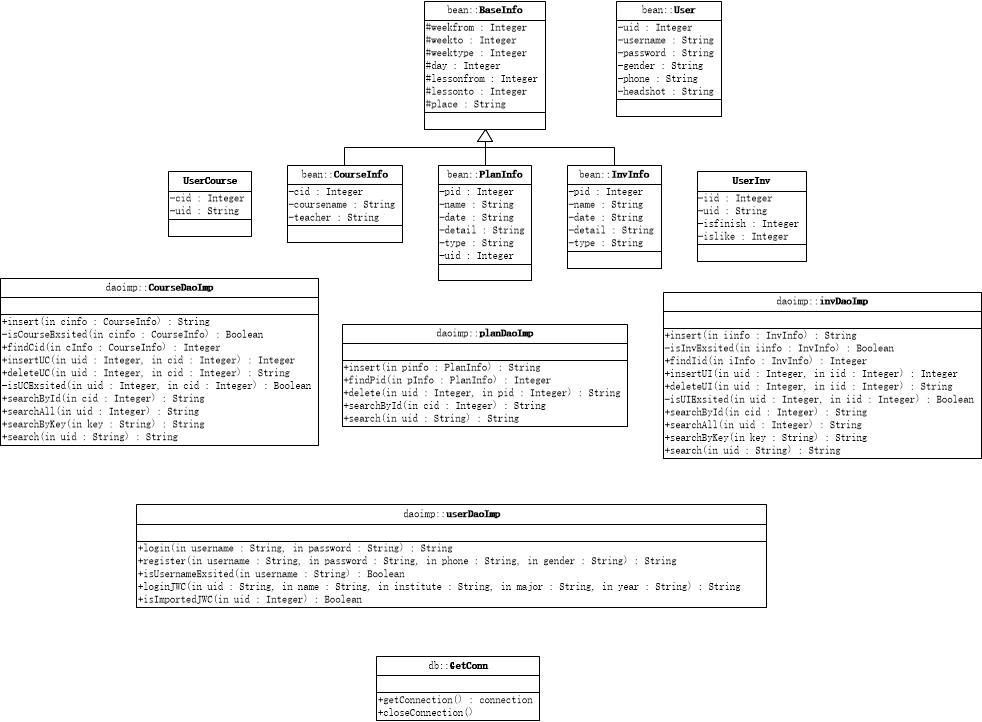


图5-5 数据处理层主要类图

Bean类，实体类，对所有属性具有getter和setter类。此处由于CourseInfo、 PlanInfo与InvInfo具有很大程度的相似度，因此将其相同属性封装为BaseInfo类，使CourseInfo、PlanInfo与InvInfo来继承BaseInfo。

DaoImp类，封装了对每张数据表的CRUD操作。

GetConn类，用于与mysql数据库进行连接。

## 5.3 本章小结

本章主要阐述了本课表规划结伴应用的设计与实现的具体细节，其中按实现模块依次介绍了客户端的详细设计，包括滑动菜单的设计实现、主界面动态显示的设计与实现、爬虫的设计与实现、以及各模块的设计与实现，并介绍了其中主要类的实现方法。同时本章还介绍了服务器端的详细设计，包括两层业务逻辑处理与相关类的主要实现方法。

# 第六章 测试与成果展示

## 6.1 测试环境

本项目开发完成后进行了客户端与服务器端的测试，测试的客户端设备有google nexus 4 等Android模拟器与Samsung S4真机。具体测试机型与配置情况如下表：

表6-1 测试机型与配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 机型 | 屏幕大小 | 分辨率 | 内存 | 系统 |
| Nexus S | 4.0 | 480\*800 | 512M | Android 5.1 x86 |
| Nexus 4 | 4.7 | 768\*1280 | 1024M | Android 7.0 x86 |
| Nexus 5 | 4.95 | 1080\*1920 | 1024M | Android 7.0 x86 |
| Samsung S4 | 5.0 | 1080\*1920 | 2G | Android 6.0 x86\_64 |

本项目服务器端采用Lenovo Y410P机型，Windows 8.1中文版64位操作系统，Intel Core i5dCPU和4G的内存环境进行测试。客户端与服务器端采取本地连接的方式进行数据传输与交换。

## 6.2 功能性测试

本应用的功能测试是采用黑盒测试技术进行的。首先使用等价类划分和边界值分析方法进行测试用例的设计，把所有可能的输入数据划分成若干子集，再从每个子集中选取一些有代表性的数据作为测试用例，从而保证用户的操作都能被应用识别并作出相应的响应，同时也验证了系统模块之间的协作性能，保证了该应用满足用户需求。下表简单列举了一些测试用例与测试结果：

表6-2 测试用例与测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块 | 前提条件 | 操作 | 预期结果 | 测试结果 |
| 主界面显示 | 登录成功 | 标题栏弹出框选择相应周数 | 正常显示该用户在本周的所有课程信息、个人规划信息与邀约信息 | 通过 |
| 滑动菜单显示 |  | 点击左上角菜单按钮或从屏幕左边向右滑动 | 唤出滑动菜单界面 | 通过 |
| 点击菜单选项 |  | 点击按钮 | 跳转到相应界面 | 通过 |
| 输入框 | 进入登录、注册、添加课程、查找课程、添加规划、发布邀约、查找邀约界面 | 点击输入框 | 输入框中光标闪动，跳出字母或数字键盘，允许输入 | 通过 |
| 滑动选择器 | 添加课程、查找课程、添加规划、发布邀约、查找邀约界面 | 点击日期、时间、地点 | 跳出滑动选择器，点击确定后，输入框内显示选择的内容；地点可以级联显示 | 通过 |

（续表6-2）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 添加与发布按钮 | 添加课程、添加规划、发布邀约界面 | 点击右上角对勾 | 提交服务器并返回添加结果，跳转主界面 | 通过 |
| 删除按钮 | 课程详细界面、规划详细界面、邀约详细界面 | 点击删除按钮 | 提交服务器并返回提示删除结果，跳转主界面 | 通过 |
| 加入按钮 | 查找课程、查找邀约界面 | 选择某项点击加入 | 提交服务器并返回加入结果，跳转主界面 | 通过 |
| 返回按钮 | 除登录、注册和主界面外其他界面 | 点击左上角返回按钮 | 返回主界面 | 通过 |
| 滑动开关 |  | 点击滑动开关 | 该项操作开启或关闭 | 通过 |
| 提醒 | 打开了提醒滑动开关 | 系统时间达到用户设置的提醒时间 | 手机消息栏弹出通知，显示即将开始的课程名或规划名、时间与地点 | 通过 |
| 上课静音 | 打开了上课静音滑动开关 | 系统时间达到用户某节课的上课时间 | 手机自动开启静音勿扰模式 | 通过 |
| 下课恢复 | 打开了上课静音滑动开关 | 系统时间在用户某节课的下课时间 | 手机自动关闭静音勿扰模式 | 通过 |

针对测试过程中发现的问题进行修改与优化后，本应用最终通过测试。

## 6.3 兼容性与性能测试

本应用开发完成后进行了功能测试，测试的设备是在google nexus 4 API 24 Android模拟器与Samsung S4真机上进行的。

兼容性与性能测试如下表：

表6-3 兼容性与性能测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 机型 | 界面情况 | 界面切换时间 | 爬取课表响应时间 | 访问服务器查询响应时间 |
| Nexus S | 正常 | <1s | 1-3s | 1-2s |
| Nexus 4 | 正常 | <1s | 1-3s | 1-2s |
| Nexus 5 | 正常 | <1s | 1-3s | 1-2s |
| Samsung S4 | 正常 | <1s | 2-3s | 2-3s |

课表爬取同时也取决于当前服务器的网络状况，以及学校教务处服务器的网络状况，因此此项测试只具有参考性。从上表可以看出，应用的兼容性和性能都不错，能适应市场上大部分主流机型，本应用通过了非功能性测试。

## 6.4 成果展示

下面将按用户使用流程来展示本应用主要功能的运行界面：

### 6.4.1 用户注册登录

|  |  |
| --- | --- |
| login.jpg | login.jpg |
| 图6-1用户登录 | 图6-2用户注册 |

### 6.4.2 课程管理

|  |  |  |
| --- | --- | --- |
| login.jpg |  |  |
| 图6-3未登录教务处 | 图6-4登录教务系统 | 图6-5导入课表后 |
|  |  |  |
| 图6-6查看详细信息 | 图6-7手动添加课程 | 图6-8添加课程成功 |
|  |  |  |
| 图6-9选择蹭课 | 图6-10关键字查找 | 图6-11蹭课成功 |

### 6.4.3 个人规划管理

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 图6-12添加个人规划 | 图6-13添加规划成功 | 图6-14规划详细信息 |

### 6.4.4 邀约管理

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| 图6-15发布邀约 | 图6-16发布成功 | | 图6-17详细信息 | |
|  |  |  | |
| 图6-18查找邀约 | 图6-19查找结果 | 图6-20加入冲突 | |
|  |  |  | |
| 图6-21邀约未完成 | 图6-22邀约未完成 | 图6-23邀约点赞 | |

### 6.4.5 提醒与上课静音

|  |  |  |
| --- | --- | --- |
|  |  | main1.jpg |
| 图6-24上课静音开启 | 图6-25上课静音 | 图6-26打开提醒 |
| main1.jpg | main1.jpg |  |
| 图6-27更改提醒时间 | 图6-28通知提醒 |  |

### 6.4.6 设置

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 图6-29设置界面 | 图6-30个人设置 | 图6-31修改密码 |

## 6.5 本章小结

本章首先介绍了Android课表应用的测试情况，进而展示了课表规划结伴应用主要功能模块的实际运行结果。应用运行情况良好，界面设计简洁，操作简单，功能相对完善，能够满足用户需求。

# 第七章 总结

本文首先分析了当前大学生的时间利用情况，并根据当下大学生们时间规划上存在的种种问题提出了相应的解决思路，同时本文还比较分析了目前市场上各种课表应用的优缺点并提出了本应用的改进方案。其次，本文简要阐述了应用中所涉及的主要技术核心。之后，本文从需求、设计、实现以及测试方面对本应用进行了详细阐述，并对实现过程中出现的难点提出了相应的解决方案。本项目基于现代软件工程理论所开发，采用传统瀑布模型，在需求分析和设计中采用了UML建模分析，使项目细节更加清晰。

本应用使用了简洁大方的UI设计，实现了课程管理、规划管理、邀约结伴、提醒、个人管理以及设置等多种功能，在帮助大学生养成时间规划习惯的同时，还建立了有效的结伴邀约监督机制，提高了规划完成的效率。尽管应用满足了目前的主要需求，但仍存在以下问题：

1）为了让四川大学本科生使用起来更为方便，本应用将上课节次、地点、教务处导入系统设计实现为适配四川大学当前设置，但因此也使得本应用具有很大的群体局限性；

2）课程数量取决于使用本应用登录教务系统爬取到课程的人数，未能实现获取全校所有课程；

3）暂时缺少新加入邀约后对已加入用户的通知推送以及用户间聊天的功能；

4）还未能建立对结伴成功后的规划执行效果的后续监督机制。

同时，该项目在实际的研发中仍存在某些技术问题有待改进，因此，还需努力学习先进技术以便后续更好地优化改善本项目。

# 参考文献

# 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

学位论文作者（签名）

论文指导教师（签名）

2017年 月 日

# 致 谢

时光飞逝，转瞬间，大学四年的时光悄悄溜过。在这四年里，有过拼搏奋斗，有过勇敢尝试，这其中离不开父母给予我的无尽鼓励与支持，离不开老师们的谆谆教诲和指导，离不开身边同学的无私帮助与照顾，更离不开在川大这四年的学习生活经历带给我的成长影响。在此，感谢远在家里的父母对我的养育之恩，感谢四川大学计算机学院所有老师对我的教育和培养，感谢辅导员侯老师对我的关怀和帮助，也感谢在校期间曾经帮助过我的所有同学。

在毕业论文完成之际，更要特别感谢我的毕业论文指导老师潘薇老师。从论文选题开始，潘老师就不辞辛苦的对我论文课题中的想法与设计进行一一指点，耐心解答我提出的每一个问题，同时还对我论文中的小问题提出系统性的建议，让我得以不断完善自己的论文。借此机会，我谨向潘老师表示衷心的感谢！

# 附录1

Testing Android Apps Through Symbolic Execution

ABSTRACT

There is a growing need for automated testing techniques aimed at Android apps. A critical challenge is the systematic generation of test cases. One method of systematically generating test cases for Java programs is symbolic execution. But applying symbolic execution tools, such as Symbolic Pathfinder (SPF), to generate test cases for Android apps is challenged by the fact that Android apps run on the Dalvik Virtual Machine (DVM) instead of JVM. In addition, Android apps are event driven and susceptible to path-divergence due to their reliance on an application development framework. This paper provides an overview of a two-pronged approach to alleviate these issues. First, we have developed a model of Android libraries in Java Pathfinder (JPF) to enable execution of Android apps in a way that addresses the issues of incompatibility with JVM and path-divergence. Second, we have leveraged program analysis techniques to correlate events with their handlers for automatically generating Android-specific drivers that simulate all valid events.

Categories and Subject Descriptors

D.2.5 [Testing and Debugging]: Symbolic Execution or Testing Tools

General Terms

Reliability, Security

Keywords

Java Pathfinder, Symbolic Pathfinder, Android, Testing

1. INTRODUCTION

In 2008, Google and Open Handset Alliance launched Android Platform for mobile devices. Android is a comprehensive software framework for mobile communication devices including smartphones and PDAs. Android has had a meteoric rise since its inception partly due to its vibrant app market that currently provisions over half a million apps, with thousands added and updated on a daily basis. Not surprisingly there is an increasing demand by developers, consumers, and market operators for automated testing techniques applicable to Android apps.

A promising automated testing technique is symbolic execution [12], a program-analysis technique that uses symbolic values, rather than actual values, as program inputs. It gathers the constraints on those values along each path of the program and with the help of a solver generates inputs for all reachable paths. Although Android apps are developed in Java, they introduce three challenges for symbolic execution tools targeted at Java, namely Symbolic PathFinder (SPF) [22].

First challenge is that Android apps depend on a proprietary set of libraries that are not available outside the device or emulator. Android code runs on Dalvik Virtual Machine (DVM) [6] instead of the traditional Java Virtual Machine (JVM). Thus, Android apps are compiled into Dalvik byte-code rather than Java byte- code. For using SPF to symbolically executed Android apps, they need to be transformed into the corresponding Java byte-code representation.

The second challenge is that Android programs’ dependence on framework libraries makes them prone to path-divergence problem, and more so than traditional Java programs. In general, path-divergence problem may occur when a symbolic value flows outside the context of the program that is being symbolically executed and to the context of the bounding framework or any external library. In Android, however, path-divergence is the norm, rather than the exception. A typical Android app is composed of multiple Activities and Services, playing the role of software components, which communicate extensively with one another using Intents, Android’s messaging system. An Intent is used to carry over a value to another Activity/Service and as a result that value leaves the boundaries of the app and is passed through Android libraries before it is retrieved in the new Activity/Service.

Finally, Android is an event driven system and extracting program input values is highly dependent on user action most of the time, meaning that the symbolic execution engine has to wait for the user to interact with the system and tap on a button or initiate some other type of event for the program to continue the execution of a certain path. Furthermore, the system itself or a third application can initiate an event and cause the app to behave in a certain way. Such events are far more frequent in smartphone apps than traditional software systems, due to the context sensitive nature of smartphones.

Current techniques for dealing with traditional event based systems either use Capture-replay or model driven approaches. In Capture-replay approaches [1, 16, 17], user records her interaction sequences with the GUI, which are replayed in time of testing. Model driven techniques [15, 24] require user to provide a model of the software system’s usages. Both Capture-replay and model driven approaches depend on manual effort, thus not very convenient, and prone to missing ways in which an app could be engaged that are not readily known. In fact, in the context of security testing, an app may intentionally incorporate hidden ways in which it can be engaged. There have also been efforts to extract directed graph models automatically by crawling the GUI [1, 16, 17], and use those graphs to generate test sequences, but again may fail to identify other ways in which a system can be engaged.

In this paper, we provide an overview of a multi-faceted approach to tackle these challenges. We describe an extension to SPF that provides stubs modeling Android libraries. These models enable us to compile Android apps on JVM, and run them on Java PathFinder (JPF) to address the first challenge, i.e., incompatibility of DVM and JVM. In addition, we provide the logic in certain stubs to simulate the behavior of Android library classes to address the problem of path-divergence. Finally, we leverage both our knowledge of Android specification and a specialized call-graph model of the app to correlate the events with their handlers [13]. Using this model we can automatically generate drivers for extracting the user input values and generating the sequence of valid events for exercising an app. Drivers address the last challenge by guiding the generation of event sequences aimed at simulating an actual user’s behavior. The resulting extended version of SPF enables us to symbolically execute Android apps to generate test cases that achieve high code coverage.

This paper is organized as follows. Section 2 provides the background on Symbolic PathFinder and Android. Section 3 presents an Android app that is used for illustrating the research. Section 4 outlines an overview of our approach. Section 5 provides a detailed view of how we model Android libraries, while Sections 6 provide the details of our approach in generating the drivers for Android apps. The paper concludes with an overview of the related research in Section 7, and a discussion of our future work in Section 8.

2. BACKGROUND

In this section we first provide a brief background on SPF,

followed by a more detailed overview of Android framework.

2.1 Symbolic PathFinder

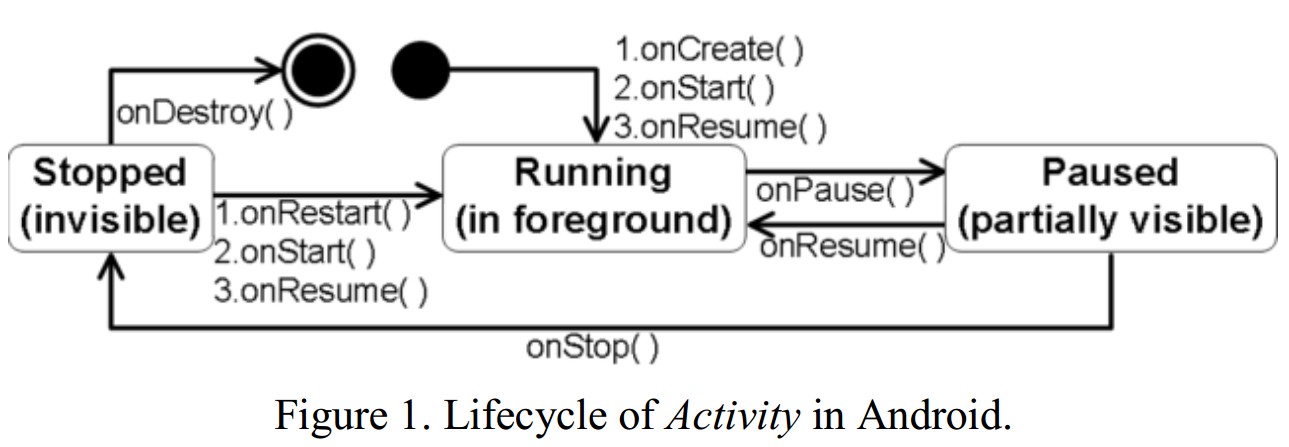
JPF [11] is a general purpose model checker for Java programs that uses its own Virtual Machine rather than traditional JVM. Symbolic Pathfinder (JPF-Symbc) is built on top of JPF as an extension implementing a non-standard interpretation of Java byte-code using a modiﬁed JPF JVM [5]. SPF analyzes Java byte- code and can handle mixed integer and real constraints, as well as complex mathematical constraints via heuristic solving. SPF can be used for test input generation and for ﬁnding counterexamples to safety properties [23]. We are extending SPF to model Android apps and leverage it to generate inputs and test cases for them.

2.2 Android

Android is a comprehensive software framework for mobile communication devices including smartphones and PDAs. The Android framework includes a full Linux operating system based on the ARM processor, system libraries, middleware, and a suite of pre-installed applications. Google Android platform is based on DVM for executing and containing programs written in Java. Android also comes with an application development framework, which provides an environment for application development and includes services for building GUI applications, data access, and other component types. The framework is designed to simplify the reuse and integration of components.

Each Android app has a mandatory manifest file. This is a required XML file for every app and provides essential information for managing the life cycle of an app in the Android platform. Examples of the kinds of information included in a manifest file are descriptions of the application’s Activities, Services, Broadcast Receivers, and Content Providers among other architectural and configuration properties.

An Activity is a screen that is presented to the user and contains a set of layouts (e.g., LinearLayout that organizes items within the screen horizontally or vertically). The layouts contain GUI controls, known as view widgets (e.g., TextView for viewing text and EditText for text inputs). The layouts and its controls are usually described in a configuration XML file with each layout and control having a unique identifier. A Service is a component that runs in the background and performs long running tasks, such as playing music. Unlike an Activity, a Service does not present the user with a screen for interaction. A Content Provider manages structured data stored on the file system or database, such as contact information. A Broadcast Receiver responds to system wide announcement messages, such as the screen has turned off or the battery is low. Activities, Services, and Broadcast Receivers are activated via Intent messages. An Intent message is an event for an action to be performed along with the data that supports that action. Intent messaging allows for late run-time binding between components, where the calls are not explicit in the code, rather connected through event messaging.



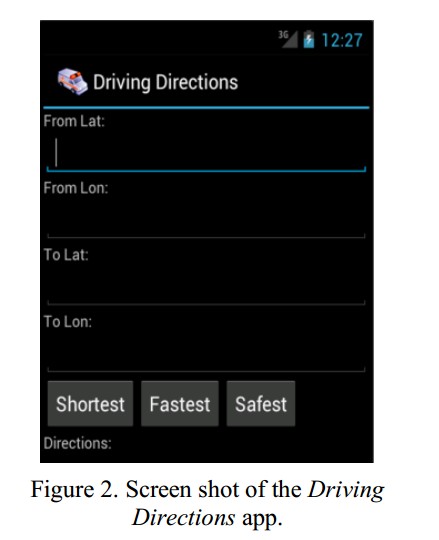
Activity and Service are required to follow pre-specified lifecycles [3]. For instance, Figure 1 shows the events in the lifecycle of an Activity: onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), and onDestroy(). These lifecycle events play an important role in our research as explained later.

In addition to these components, a typical application utilizes many resources. These resources include animation files, graphics files, layout files, menu files, string constants, styles for user interface controls. Most of these are described using XML files. An example, as mentioned before are layouts. The layout XML files define the user interface controls that are used by Activities. The resources each have a unique identifier that is used to distinguish and get a reference to them in the application code.

3. ILLUSTRATIVE EXAMPLE

For illustrating the approach, we will use a Driving Directions app, a subset of a software system, called Emergency Deployment System (EDS) [14]. EDS is intended to allow a search and rescue crew to share and obtain an assessment of the situation in real-time (e.g., interactive overlay on maps), coordinate with one another (e.g., send reports, chat, and share video streams), and engage the headquarters (e.g., request resources).

Driving Directions app can be used to calculate off-road driving directions between two geographic points, while considering cost objectives such as distance, time, and safety. It also provides the user with the closest dispatches to source, destination or on the route. Figure 2 depicts the GUI of this Android app. The input boxes are for latitude/longitude pair and the buttons for alternative ways of computing the directions. The latitude/longitude coordinates can be typed in or selected from a map. The resulting turn-by-turn directions are shown in a separate text box, and optionally displayed on a map.



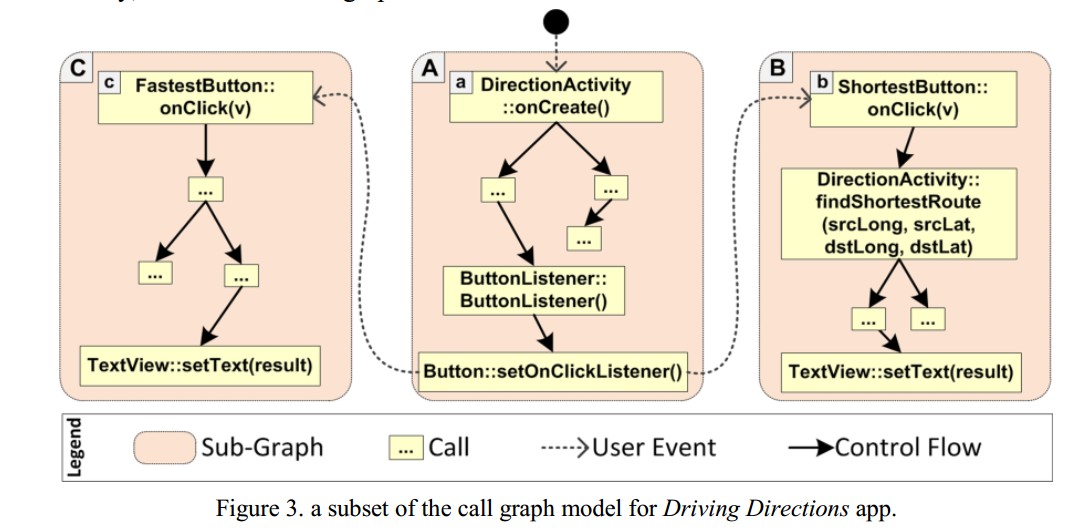
4. APPROACH OVERVIEW

The main objective of our tool is to automatically generate test inputs for Android apps using SPF. In order to do so, we first need to generate the Java byte-code of the app that can be executed on JVM. Hence, we have to compile the app’s source code with Java compiler, instead of Android’s Software Development Kit (SDK). This is achieved by first replacing platform-specific parts of the Android libraries that are needed for each app with our models. These models are essentially stubs that are created in a way that each component’s composition and callback behavior is preserved. This allows us to execute an Android app on JPF without modifying the app’s implementation.

In addition, we have to take into account the fact that a typical app is composed of multiple components (i.e., Activities and Services) that communicate using Android’s Intent messages. Path- divergence issue arises when an Intent is used to pass a symbolic value to another Activity or Service. To solve this problem, we have implemented the appropriate logic for simulating how Android platform mediates the event-based interaction of components, and incorporated that logic in the Activity, Service, and Intent stubs. We refer to these enhanced stubs as mock classes, as they emulate how the corresponding Android constructs behave. This way we are able to address the path- divergence problem caused by the basic mechanism in which Android components communicate with one another.

The second step in our approach is to generate drivers for an app using its call graph model. Unlike traditional Java programs, Android apps do not contain a main class that becomes the root node of the call graph, where the program is always initiated. Android apps are event driven, meaning that the thread of execution constantly changes context between the application logic, system, and user. Therefore, instead of a connected call graph that represents the complete control flow of the application, an Android app is composed of a set of disconnected sub-call graphs that collectively represent the app’s logic. These sub-call graphs correspond to all the ways in which an app can be initiated, accessed by the user or Android platform. Figure 3 illustrates a hypothetical call graph model for an Android app, where A, B and C are each a sub-call graph with root nodes a, b, and c, while the black circle represents the start of the app.

Following the generation of the sub-call graphs, we parse the source code, the resources and configuration information, including the manifest file, to find the event handlers. This allows us to automatically connect sub-call graphs in order to construct the call graph model of the system. Figure 3 illustrates a subset of the Driving Directions app’s call graph model. The dotted arrows that connect two sub-graphs illustrate the initiation of an event either by the user or the system. The call graph model represents all possible method invocation sequences (execution traces) within an app, which is depicted by black arrows in the figure.



Finally, we use the call graph model to derive a Context Free Grammar (CFG) that is used to generate drivers. A driver is a sequence of events that simulate user’s interaction with the app. Since the root node of each sub-graph represents an event being handled, every valid combination of the root nodes represents a scenario of user interaction with the app. For instance, based on the call graph model in Figure 3, {a,c} and {a,c,b} are two possible event sequences. We monitor the code coverage from the execution of tests using EMMA [7], an open source toolkit that monitors and reports Java code coverage. We start with a single event driver and continue generating drivers with more event sequences iteratively. The testing stops when we hit a pre-specified code coverage threshold. We describe the approach in more detail in the next two sections.

5. MODELING ANDROID LIBRARIES

The conventional technique to tackle the path-divergence problem, which occurs when a Java program depends on external libraries, is to provide stubs for those libraries. We explored three possible techniques for developing stubs for supporting Android apps as describe below.

The first and probably the most straightforward approach for us would have been to leverage the implementation of Android’s library classes. Android platform provides a set of library classes that apps use to access the resources on the phone. These library classes contain native methods and Java Native Interface (JNI) calls that are platform dependent and cannot be executed outside of an actual phone. To support the development activity, Google provides android.jar with Android’s Software Development Kit, which allows developers to resolve the dependencies and compile apps in the development environment. At first blush, it may seem that this jar file could be used to execute apps on top of JPF. However, upon further inspection, we found that Google has stripped the library classes in this jar file, and replaced all the method bodies to throw an exception.

A second approach we explored was similar to that employed in Robolectric [20], a framework for running Android unit tests outside of Android emulator and on top of JVM. In this approach, one would use shadow classes, which are mock classes that are accessed through reflection. Shadow classes simulate the behavior of the actual Android library classes, and override the calls to them through reflection. This approach may be useful for running test cases on JVM, but not feasible for use with JPF, because currently JPF is not capable of handling reflection.

Finally, a third approach, and the one that we have adopted in our implementation is to provide our own custom built stub and mock classes. The stub classes are used to compile Android apps into JVM byte-code, while mock classes are used to deal with the path-divergence problem. We developed stubs that return random values, when the return type of a method is primitive, and return empty instances of the object, when the return type is a complex data type.

Dealing with Android platform, not only we need to provide stub classes to resolve the byte-code incompatibility with JVM, but we also need to address the lack of Android logic outside the phone environment. Android uses its library classes as bolts and nuts that connect the different pieces of an app together. For example, and as shown in Listing 1, in Driving Directions app, DirectionsActivity uses the startActivity method of the Android library class Activity.java to start the app’s DisplayRouteActivity that displays the shortest route. It creates an Intent in which the source and destination activities along with the values to be carried are specified. In this case we provide the appropriate logic for startActivity mock, such that when a new instance of DisplayRouteAcitivity is created the control flow moves to its onCreate method.

Moreover, we create a mock for the Intent.class to address the path-divergence problem. As shown in Listing 1, an instance of Intent is passed to startActivity. This Intent encapsulates symbolic values of sourceLong, sourceLat, destLong and destLat, and causes path-divergence. To deal with this issue, we provided our own implementation of putExtra and getExtra methods in the mock implementation of Intent.java, such that the symbolic value of those variables is preserved. Android uses a hashMap<String, Object> to store and retrieve the values stored in an Intent, making it difficult to reason about a value stored as Object symbolically. To solve this problem, we provide our model of a hash map that holds primitive values. Consequently, in our implementation of the putExtra and getExtra methods we use our own model of hash map to enable JPF to symbolically reason about values that are exchanged using the Intent messages.

# 附录2

通过符号执行测试Android应用

摘要

目前，针对Android应用的自动化测试技术的需求越来越旺盛，系统性的生成测试用例成为了一个关键性的挑战。为Java程序系统性的生成测试用例的其中一种方法就是符号测试，但应用例如符号路径查找器Symbolic Pathfinder （SPF）此类的符号执行工具去为Android应用生成测试用例受到了Android运行的虚拟环境Dalvik Virtual Machine （DVM）的影响，而非受到Java虚拟机（JVM）的影响。而且， Android应用是事件驱动的，并且因为它们对一个应用开发框架的依赖使得易出现路径分歧。本文介绍了一种双管齐下的办法来减轻这些问题。首先，我们在Java路径查找器Java Pathfinder（JPF）中开发了一个Android库模型，来用一种解决与JVM不兼容的问题和路径分歧的问题的方式执行Android应用；其次，我们和为Android特定驱动而自动生成的处理程序一起，利用程序分析技术来关联事件，模拟所有有效事件。

类别和主题描述符

D.2.5 [ 测试和调试 ]：符号执行或测试工具

普通术语

可靠性，安全性

关键词

Java符号路径查找器，符号路径查找器，Android，测试

1介绍

2008年，Google和开放手机联盟一起为移动设备推出了Android平台。 Android是一个包括智能手机和PDA在内的移动通信设备的综合软件框架。起初部分是由于它为超过五十万个应用程序、以及每天数千个新增和更新的应用程序提供了一个充满活力的应用程序市场，因而Android才有了一个飞速的增长。不出意料的是，开发人员、消费者和市场运营商对Android应用程序的自动测试技术的需求日益增加。

一个有前途的自动测试技术是符号执行[12]，这是一种使用符号值而非实际值作为程序输入的程序分析技术，它聚集了程序的每条路径上的值的限制，并且在解算机的帮助下，为所有可达路径生成输入。虽然Android应用程序是用Java开发的，但它引入了面向Java的符号执行工具，即符号路径查找器（SPF）[22]所面临三大挑战。

第一个挑战是Android应用程序依赖于一组专有库，这些库在设备或模拟器之外是不可用的。Android代码是 运行在Dalvik虚拟机（DVM） [6]上的，而不是传统 的Java虚拟机（JVM）。因此，Android的应用程序被编译成Dalvik字节码而不是Java字节码。 为了使用SPF来符号执行Android应用，就需要转换成相应的Java字节码来表示。

第二个挑战是Android程序对框架库的依赖使它们比传统的Java程序更容易出现路径分歧的问题。 一般而言，当符号值在符号执行的程序上下文之外或是在边界框架或任何外部库上下文流动时，路径分歧问题就可能会发生。然而，在Android中，路径分歧是正常的，而不是异常。 一个典型的Android应用是由多重活动Activity和服务Service组成的，扮演着软件组件的角色，而另一个使用Android的邮件系统Intent与之进行通信，Intent是用于将一个值传递到另一个 活动Activity或服务Service，并且导致该值离开该应用的边界并在恢复新的活动Activity或服务Service之前就通过Android库被传递了。

最后，Android是一个事件驱动的系统，并且大部分时间提取程序输入值是要高度依赖用户操作，这意味着符号执行引擎必须等待用户与系统进行交互，然后点击按钮或为程序启动一些其他类型的事件来继续执行某一确定的路径。 此外，系统本身或是第三方应用程序可以启动一个事件并引导该应用程序以某种方式运行。相比于传统的软件系统，这样的事件在智能手机的应用程序中更为频繁，因为智能手机上下文敏感的性质。

目前基于系统而用于处理传统事件的技术不是使用捕获重放的方法就是使用模型驱动方法。在捕获重放方法[1,16,17]，用户记录了她的和GUI的交互次序，在测试时就被重放了出来；模型驱动技术[15,24]要求用户提供一个软件系统使用方法的模型。捕获重放和模型驱动方法都取决于人力，因此不是非常方便，并且易于出现应用程序进入不易被发现的错误中。事实上，在安全测试的上下文中，一个应用程序可能会故意包含其中可以参与的隐藏方法。同时颇费周折的是，提取通过爬行GUI自动生成有向图模型[1，16，17]，并使用这些图生成测试序列，但系统参与的其他方式的识别也可能再次失败。

在本文中，我们概述了一个多方面的方法来应对这些挑战。我们描述了提供建立Android库的桩SPF的扩展。这些模型使我们在JVM上编译Android应用程序，并在Java路径查找器（JPF）上执行程序来解决第一个挑战，即DVM与JVM不兼容。另外，我们提供特定桩的逻辑来模拟Android库中类的行为来解决路径分歧的问题。最后，我们利用我们的Android规范和应用程序的专用调用图模型来与他们的处理程序[13]关联起来。使用这个模型我们可以自动地生成用于提取用户输入值和产生运行中的应用程序的有效事件序列的驱动程序。驱动程序通过引导目标是模拟实际用户的行为的事件序列的产生解决了最后的挑战。所产生的SPF的扩展版本使我们能够符号执行Android应用程序来生成实现高代码覆盖面的测试用例。

本文的组织结构如下:第2节介绍了符号路径查找器和Android的背景，第3节举了一个Android应用程序的例子，用来说明我们的研究，第4节概述了我们的方法，第5节介绍了我们如何建模Android库的详细视图，而第6节为Android应用生成驱动程序的方法提供了细节，本文在第7节总结了相关研究的概述，并在第8节讨论了未来的前景。

2背景

在这一节我们首先对SPF做一个简短的背景介绍，其次是对Android框架的一个更详细的概述。

2.1符号路径查找器Symbolic PathFinder

JPF [11]是Java程序的通用模型检查器，它使用自己的虚拟机而不是传统的JVM。符号路径查找器（JPF-Symbc）是作为一个扩展实现了修正过的JPF JVM [5]的Java字节码的非标准解释。 SPF分析Java字节码，并且可以处理混合的整数和实际约束，以及通过启发式求解处理复杂的数学约束。 SPF可以用于测试输入生成，也可以用于找出安全属性[23]的反例。 我们正在扩展SPF为Android应用程序建模，并利用它来为它们生成输入和测试用例。

2.2 Android

Android是一个包括智能手机和PDA在内的移动通信设备的综合软件框架。Android框架包括一个完整的基于ARM处理器、系统库、中间件和预安装应用程序套件的Linux操作系统。 Google Android平台是基于用于执行和包含以Java编写的程序的DVM。Android的还配备了一个 应用程序开发框架 ，这为应用程序开发以及用于构建GUI应用程序、数据访问和其他组件提供了一个环境。该框架旨在简化组件的重用和集成。

每个Android应用程序都有一个强制执行的 清单文件。 这是一个每个应用程序都必需的XML文件，它为管理Android平台上的应用程序的生命周期提供必要的信息。包含在清单文件中的信息种类的例子有在其他架构和配置属性之中的应用程序的活动、服务Service、广播接收器Broadcast Receivers和内容提供器Content Provider。

活动Activity是一个呈现给用户，并且包含一组布局的屏幕（例如，LinearLayout可以在屏幕中水平或垂直地组织项目）。布局包含被称为 视图控件的GUI控件（例如，TextView用于查看文本，EditText用于文本输入）。布局及其控件通常在配置XML文件中描述，每个布局和控件都具有唯一的标识符。A 服务是一个在后台运行并执行长时间运行任务的组件，例如播放音乐这样长时间的运行任务。与活动不同，服务不会向用户展示交互屏幕。内容提供器负责管理存储在文件系统或数据库上的结构化数据，例如联系人信息。广播接收器响应系统范围的通知消息，例如屏幕已关闭或电池电量不足。活动，服务和广播接收器通过Intent消息被激活，Intent消息是与支持该操作的数据一起执行操作的事件。Intent消息允许组件之间延迟运行的绑定，其中在代码中调用不是显式的，而是通过事件消息传递进行调用。

活动和服务必须遵守预先指定的生命周期[3]。例如，图1展示了活动的生命周期中的事件：onCreate()，onStart()，onResume()，onPause()，onStop()，onRestart()，和onDestroy()。正如下文所述，这些生命周期事件在我们的研究中发挥着重要作用。

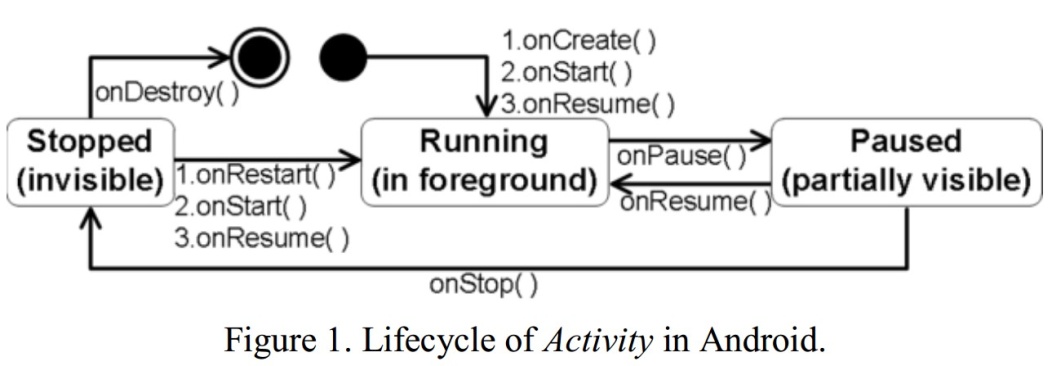


图1Android中活动Activity的生命周期

除了这些组件之外，典型的应用利用了很多资源。这些资源包括动画文件，图形文件，布局文件，菜单文件，字符串常量，用户界面控件样式。大多数使用XML文件来进行描述。举一个例子，正如前文所讲的布局。布局XML文件定义了被活动使用的用户界面控件。这些资源每个都有唯一的标识符，用于在应用程序代码中区分并获取对它们的引用。

3 说明性例子

为了说明我们的方法，我们将举一个行车指南Driving Directions应用程序的例子，这是一个软件系统的子集，名叫紧急部署系统（EDS）[14]。EDS旨在让搜救队员分享并实时获得有关情况的评估（例如，在地图上交互叠加），相互协调（例如，发送报告，聊天和共享视频流）并参与总部（例如，请求资源）。

行车指南应用可用于计算两个地理位置之间的越野驾驶方向，同时考虑成本目标如距离，时间和安全。它还为用户提供最接近出发地，目的地或中间地址的调度。图2展示了此Android应用的GUI。输入框用于纬度/经度对以及用于计算方向的替代方式的按钮。纬度/经度坐标可以在地图中输入或选择。所得到的逐个转弯方向显示在单独的文本框中，并可以选择性地显示在地图上。

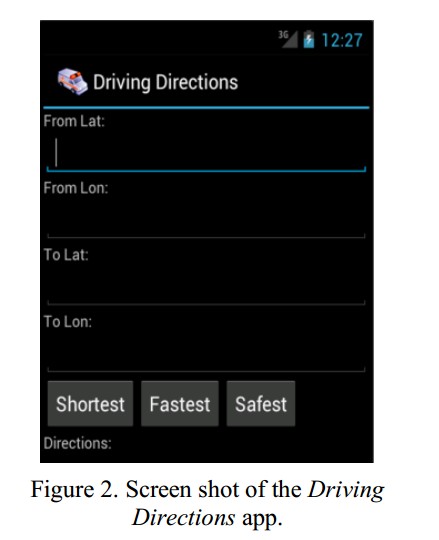


图2 行车指南应用的截图

4方法概述

我们的工具的主要目标是使用SPF自动生成Android应用的测试输入。为了做到这一点，我们首先需要生成可以在JVM上执行的应用程序的Java字节码。因此，我们必须用Java编译器编译该应用程序的源代码，而不是用Android软件开发工具包（SDK）编译。这首先是通过用我们的模型替换每个应用程序所需的Android库的平台专用部分来实现的。这些模型本质上是以保留每个组件的组合和回调行为的方式创建的桩，使得我们不修改应用程序的实现就可以在JPF上执行一个Android应用程序。

另外，我们必须考虑到这个事实，就是典型的应用程序是由使用Android的Intent的消息通信的多个部件（即活动和服务）所组成的。当Intent被用于将符号值传递给另一个活动或服务时，就会出现路径分歧问题。为了解决这个问题，我们已经实现了适当的逻辑，用于模拟Android平台如何调解组件基于事件的交互，并将该逻辑并入活动，服务和Intent的桩中。我们将这些增强的桩称为模拟类mock，因为它们模拟了相应的Android是如何构造的行为。这样我们可以解决由Android组件相互通信非基本机制所造成的路径分歧问题。

我们方法的第二步是通过使用其调用图模型为应用程序生成驱动程序。与传统的Java程序不同，当程序初始化时，Android应用程序不包含会成为调用图根节点的主类main。Android应用程序是事件驱动的，意味着执行的线程会不断更改应用逻辑、系统和用户之间的上下文环境。因此，取代一个代表应用程序完整控制流的连接的呼叫图，Android应用程序由一组断开的子通话图组成，由它们共同表示应用程序的逻辑。这些子调用图对应于由用户或Android平台访问的应用程序启动的所有方式。图3说明了对一个Android应用的假设调用图模型，其中 A，B和C各自对应子调用图上的根节点a，b和c，而黑圈代表应用程序的开始。

在生成子调用图之后，我们解析源代码、资源和配置信息，包括清单文件，去寻找事件处理程序。这使得我们可以自动连接子调用图来构造系统的调用图模型。图3说明了行车指南应用程序调用图模型的一个子集。连接两个子图的虚线箭头表示由用户或系统发起的事件的启动。调用图模型表示应用程序中的所有可能的方法调用序列（执行跟踪），并由图中的黑色箭头所描绘。

最后，我们使用调用图模型来导出用来生成驱动程序的上下文无关语法（CFG）。驱动程序是模拟用户与应用程序交互的一系列事件。由于每个子图的根节点表示正在处理的事件，根节点的每个有效组合表示用户与应用程序交互的场景。例如，基于图3中的调用图模型，{a，c}和{a，c，b}是两个可能的事件序列。

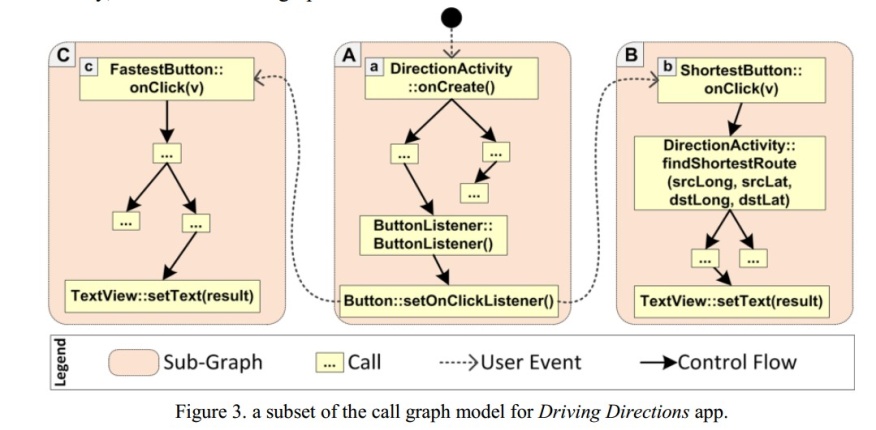


图3行车指南应用调用图模型的子集

我们使用EMMA [7] 来监视执行测试的代码覆盖率，它是一个监视和报告Java代码覆盖率的开源工具包。我们从单个事件驱动开始，并继续以更多事件序列迭代生成驱动程序。当我们达到预先指定的代码覆盖率阈值时，测试便会停止。我们将在接下来的两节中更详细地描述该方法。

5 Android库的建模

解决Java程序依赖外部库时发生的路径分歧问题的常见技术是为这些库提供桩。我们探讨了三种可能的技术来开发支持Android应用程序的桩，如下所述。

第一个也可能是最直接的方法是利用Android库类的实现。 Android平台提供了一套库类，使应用程序可以使用它去访问手机上的资源。这些库类包含本地方法和依赖于平台的Java本地接口Java Native Interface（JNI）的调用，它不能在实际的手机之外执行。为了支持开发活动，Google提供的android.jar中带有Android的软件开发工具包，这允许开发人员解析依赖并在开发环境中编译应用程序。起初乍一看，这个jar文件似乎可以用于在JPF之上执行应用程序。不过，经过进一步的检查，我们发现Google已经有了剥离这个jar文件中的库类，并替换所有的方法体来抛出异常。

我们探索的第二种方法类似于Robolectric [20]中采用的方法，它是在Android模拟器之外运行Android单元测试并在JVM之上运行的框架。在这种方法中会使用阴影类shadow，它们是通过镜像访问的模拟类。阴影类模拟实际的Android库类的行为，并通过镜像来覆盖对它们的调用。这种方法对于在JVM上运行测试用例可能是有用的，但是不可用于JPF，因为目前的JPF不能处理镜像。

最后，第三种方法，我们在实现中采用的方法是使用我们自己定制的桩类stub和模拟类mock。桩类是用来将Android应用程序编译为JVM字节码，而使用模拟类处理路径分歧问题。当方法的返回类型是原始的时，我们开发了能返回随机值的桩；并且当返回类型是复杂数据类型时，就要返回对象的空实例。

处理Android平台，不仅需要提供桩类来解决与JVM不兼容的字节码，而且还需要解决手机环境之外的Android逻辑缺陷。Android将其库类作为将应用程序的不同部分连接在一起的螺栓和螺母。例如，正如表1中所表明的一样，在行车指南应用程序中，DirectionsActivity所使用Android库类 Activity.java中的startActivity方法来启动本应用中显示最短路径的活动DisplayRouteActivity。它创建一个Intent，其中指定源活动和目标活动以及要传递的值。在这种情况下，我们提供了适用于startActivity模拟的逻辑，这样当创建了一个新的DisplayRouteAcitivity实例时，控制流将移动到其onCreate方法中。

此外，我们为Intent.class创建了一个模拟来解决路径发散问题。 如表1所示，Intent的实例传递给startActivity，此Intent封装了sourceLong，sourceLat，destLong和destLat的符号值，导致了路径分歧。为了解决这个问题，我们在Intent.java的模拟实现中提供了我们自己的putExtra和getExtra方法的实现，以便保留这些变量的符号值。 Android使用一个hashMap <String，Object>来存储和检索存储在Intent中的值，使得很难符号化地推出存储为Object的值。为了解决这个问题，我们提供了一个包含原始值的哈希映射模型。因此，在我们实现putExtra和getExtra方法时，我们使用我们自己的哈希映射模型来使JPF能够符号化地推出Intent消息交换的值。

1. [] Android程序框架设计 [EB/OL]. http://blog.csdn.net/leehong2005/article/details/8535306 [↑](#endnote-ref-2)
2. [] Android视图框架 [EB/OL]. http://www.cnblogs.com/AcerWang/p/4080318.html [↑](#endnote-ref-3)