<**打车软件系统** >

体系结构设计说明书

<1.0>

<2019.12.18>

<朱豫>

<2017211884>

<软件学院2017211502班>

# 版本历史

**日期 描述 作者 注释**

|  |  |  |  |
| --- | --- | --- | --- |
| 2019/12/18 | 版本1 | 朱豫 | 第一个版本 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 文件许可

以下的软件需求规格已被接受并认可

**签名 名称 标题 日期**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[版本历史 2](#_Toc27064666)

[文件许可 2](#_Toc27064667)

[1. 项目及文档相关信息 5](#_Toc27064668)

[1.1项目背景 5](#_Toc27064669)

[1.2文档范围 5](#_Toc27064670)

[1.2.1待开发软件系统名称 5](#_Toc27064671)

[1.2.2项目简介 5](#_Toc27064672)

[1.2.3项目价值 5](#_Toc27064673)

[1.2.4项目特色 6](#_Toc27064674)

[1.3定义、术语和缩写语 6](#_Toc27064675)

[1.4参考文献 8](#_Toc27064676)

[1.5文档概述 8](#_Toc27064677)

[2.体系结构需求 8](#_Toc27064678)

[2.1关键指标 8](#_Toc27064679)

[2.2体系结构用例 9](#_Toc27064680)

[2.3各相关方对体系结构的要求 10](#_Toc27064681)

[2.4约束条件 10](#_Toc27064682)

[2.5非功能需求 11](#_Toc27064683)

[2.6风险 11](#_Toc27064684)

[3.解决方案 12](#_Toc27064685)

[3.1相关体系结构 12](#_Toc27064686)

[3.1.1B/S体系架构 12](#_Toc27064687)

[3.1.2Broker体系架构 13](#_Toc27064688)

[3.1.3 MCV模式 14](#_Toc27064689)

[3.2体系结构概述 15](#_Toc27064690)

[3.3结构化视图 15](#_Toc27064691)

[3.3.1概念级体系结构 15](#_Toc27064692)

[3.3.2模块级体系结构 16](#_Toc27064693)

[4.用例视图 18](#_Toc27064694)

[4.1出行乘客用户 19](#_Toc27064695)

[4.2司机用户 19](#_Toc27064696)

[4.3系统检测管理员 20](#_Toc27064697)

[5.逻辑视图 22](#_Toc27064700)

[6.进程视图 23](#_Toc27064701)

[6.1进程视图 24](#_Toc27064702)

[6.1.1 出行乘客用户发布订单 24](#_Toc27064703)

[6.1.2 出行乘客用户查询订单 24](#_Toc27064704)

[6.1.3 出行乘客用户反馈问题 24](#_Toc27064705)

[6.1.4 司机用户接单 25](#_Toc27064706)

[6.1.5 司机用户查询订单 25](#_Toc27064707)

[6.1.6 司机用户反馈问题 25](#_Toc27064708)

[6.1.7 司机用户提交信息 25](#_Toc27064709)

[6.1.8 系统检测管理员查询信息 25](#_Toc27064710)

[6.1.9 系统检测管理员发布信息 25](#_Toc27064711)

[6.2设计元素的过程 26](#_Toc27064714)

[6.2.1发布订单 26](#_Toc27064715)

[6.2.2 接单 26](#_Toc27064716)

[6.2.3 查询订单 26](#_Toc27064717)

[6.2.4 发布信息 26](#_Toc27064718)

[6.2.5 审核信息 27](#_Toc27064719)

[6.2.6 修改权限 27](#_Toc27064720)

[6.3过程模型设计模型的依赖关系 27](#_Toc27064721)

[6.4实施过程 28](#_Toc27064722)

[6.4.1远程 28](#_Toc27064723)

[6.4.2可运行 28](#_Toc27064724)

[6.4.3线程 29](#_Toc27064725)

[7.部署视图 29](#_Toc27064726)

[7.1手机 29](#_Toc27064727)

[7.2计算机 29](#_Toc27064728)

[7.3 注册服务器 30](#_Toc27064729)

[7.4 数据库 30](#_Toc27064730)

[7.5 智能分析系统 30](#_Toc27064731)

[8.性能 30](#_Toc27064732)

[8.1时间性能： 30](#_Toc27064733)

[8.2可靠性 30](#_Toc27064734)

[8.3易用性 30](#_Toc27064735)

[8.4密安性 31](#_Toc27064736)

[8.5可维护性 31](#_Toc27064737)

[8.6可移植性 31](#_Toc27064738)

[8.7可扩展性 32](#_Toc27064739)

[9.系统的质量分析和评价 32](#_Toc27064740)

[9.1场景分析 32](#_Toc27064741)

[9.1.1用例场景 32](#_Toc27064742)

[9.1.2增长性场景 33](#_Toc27064743)

[9.1.3探索性场景 33](#_Toc27064744)

[9.2原型分析 33](#_Toc27064745)

[9.2.1原型评价法 33](#_Toc27064746)

[9.2.2效用树法 34](#_Toc27064747)

[9.3风险 34](#_Toc27064748)

[A．附录 35](#_Toc27064749)

# 项目及文档相关信息

## 1.1项目背景

现如今，随着社会发展与科技进步，打车软件已经进入了我们的生活中，如滴滴出行、美团打车、高德打车等。

本体系结构设计文档是在对打车软件进行了全面细致的需求分析，明确了所要开发的系统应具有的功能、性能之后编写的文档，旨在阐述打车软件的总体结构，包括逻辑设计、物理结构，分析系统的体系结构需求，包括约束条件、设计遵循的标准、非功能性需求，给出体系结构设计的解决方案并分析建模，最后进行体系结构的质量分析和评估。

本体系结构设计文档作为产品立项和产品开发的参考文档，给出各用户详细的功能要求，系统功能块组成及联系，进程部署和硬件要求等，有益于提高软件开发过程中的能见度，便于软件开发过程中的控制与管理。此体系结构设计文档是进行软件项目设计开发的基础，也是编写测试用例和进行系统测试的主要依据，它对开发的后续阶段性工作起着指导作用。同时此文档也可作为软件用户、软件客户、开发人员等各方进行软件项目沟通的基础。

本文档的预期读者对象为：

1. 开发人员：根据本文档了解预期项目的功能，并据此进行系统设计与开发。
2. 项目经理：根据体系结构定义的构件结构制定项目的开发计划。
3. 测试人员：根据体系结构设计系统的总体测试框架。
4. 维护人员：根据本文档中确定的体系结构进行软件系统维护。
5. 用户：了解预期项目的功能和性能与整体结构。
6. 其他相关人员：如用户文档编写者、项目管理人员等。

## 1.2文档范围

### 1.2.1待开发软件系统名称

打车软件系统

### 1.2.2项目简介

该软件产品是对当前城市打车状况和现有打车软件进行分析后设计的功能更加全面的手机打车软件。该项目主要是由智能分析系统进行管理，与出行乘客用户、司机用户、系统检测管理员即时交互。支持用户发布订单并查询订单相关信息，支持司机用户接单，支持系统检测管理员查看订单信息和发布指令，支持管理人员管理相关信息。

### 1.2.3项目价值

该系统的优点在于：

a、方便出行乘客用户，出行乘客用户可以实时发布订单，并且支持预约服务。出行乘客用户将订单发布在软件上，就会有司机用户酌情接单，避免了传统打车方式中附近没有出租车路过或路过的出租车有客就打不到车的情况，方便了人们的日常出行。

b、方便司机用户；以前司机用户只能通过经验在客流量大的路段等待乘客打车，无法得知哪里有订单，会浪费很多时间和精力。现在通过该软件，司机用户可以在线上实时接单，减少了等待的时间，提高了收入。

c、方便系统检测管理员，系统检测管理员可以与出行乘客用户和司机用户沟通，对服务过程中出现的问题及时进行反馈和更正。

### 1.2.4项目特色

该系统可以即时地对出行乘客用户的信息进行汇总，通过智能化和信息化的方式让司机用户接送乘客，采用智能化打车信息管理平台、打车热力大数据分析、APP 线上支付等方式，提高乘客打车效率和司机用户接送乘客效率。可以实现的功能有：

(1) 出行乘客用户：

* 发布打车信息：上车地址，目的地址，选择的车辆类型，相应的预计价格。
* 查询车辆信息：查看接单司机用户的车辆基本信息，如车辆型号、颜色、车牌号，司机用户姓名、编号和联络方式。
* 设置个人信息：个人账号名，个人手机号，绑定微信，支付宝等。
* 预定车辆。

(2) 司机用户：

* 接收订单：根据系统通知，接收附近的打车信息，确认订单。
* 查询个人信息：个人手机号，绑定微信，绑定支付宝等。
* 设置基本信息：车辆型号、颜色、车牌号码，个人姓名、公司编号和个人手机号。
* 查看打车热力图：某时间段某地区的打车状况分析图。
* 接受车辆预定。

(3) 系统检测管理员：

* 查询个人信息：司机用户信息和乘客信息等。
* 可随时查询车辆载客记录。
* 对乘客的请求进行核实并处理。
* 可对乘客和司机用户发布通知。

## 1.3定义、术语和缩写语

1、用户角色：用户角色是按照一定参考体系划分的用户类型，是能够代表某种用户特征、便于统一描述的多用户个体的集合。

2、功能性需求：功能性需求规定开发人员必须在产品中实现的软件功能，用户利用这些功能来完成任务，满足业务需求。

3、非功能性需求：非功能性需求是指依一些条件判断系统运行情形或其特性，而不是针对系统特定行为的需求。

4、功能结构图：功能结构图是将系统的功能进行分解，按功能从属关系表示的图表。

5、数据流图（DFD）：数据流图从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具，是用于表示软件模型的一种图示方法。

6、用例图：用例图是指由参与者（Actor）、用例（Use Case）以及它们之间的关系构成的用于描述系统功能的静态视图。

7、UML：统一建模语言（Unified Modeling Language）的缩写，是一种建模语言。

8、E-R图：实体-联系图提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。

9、环境图：周围环境图表示系统的范围以及与外界的交互关系。

10、B/S结构：B/S结构（Browser/Server，浏览器/服务器模式），是WEB兴起后的一种网络结构模式，WEB浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。

11、三层B/S结构：三层B/S体系结构是在B/S结构的基础上，在数据管理层(Server)和用户界面层(Client)增加了一层结构，称为中间件(Middleware)，使整个体系结构成为三层。

## 1.4参考文献

*[1]王安生.软件工程化[M].北京:清华大学出版社.2014.p164-p212.*

## 1.5文档概述

本文主要针对打车软件的软件结构进行设计，得出了打车软件的体系结构说明书，主要内容包括四部分，具体如下:

第一部分主要介绍了本文档的编写目的、使用范围以及本文使用到的定义、术语与缩写词，同时也列出了本文的参考文献。

第二部分主要对本体系结构进行了总体的概述，简述了体系结构的需求以及用例，并对本系统的约束与假设和依赖做了简单陈述。

第三部分主要介绍了对本系统的体系结构进行了分级设计。对相关的体系结构进行了 概述，并用结构化的视图描绘了概念级、模块级、以及运行级的体系结构设计，同时本系统还使用了“4+1”体系结构模型对体系结构系统进行了描绘。

第四部分对系统进行了质量分析与评价。主要使用了场景分析的方式，描绘了用例场景，增长性场景和探索性场景；并使用该场景进行手工测试。

# 2.体系结构需求

## 2.1关键指标

本打车软件是一个可以为出行提供方便的生活类APP。打车软件旨在解决人们出行打车等车时间不确定，空闲司机用户无法感知到用户出行需求的这一问题，通过使用打车软件不仅节约了乘客出行时的等车时间，还为司机用户的接客工作带来了更高的效率。

本打车软件可以给出行乘客提供打车服务的平台，可以让用户乘客无需在道路上实时打车，而是可以提前预约出行时间、预约出行地点，在恶劣天气和紧急情况出现时，此功能极大地为用户提供了方便，还能提升幸福感。

与此同时，本打车软件也为司机用户的工作提高了效率。司机用户无需在道路上循环行驶来确定需要打车的乘客然后为他们提供服务，本打车软件可以让司机用户第一时间收到附近有出行需求的用户信息，避免了司机用户自己去寻找目标乘客的麻烦，极大地提高了效率*。为了保证司机用户用户与乘客用户在短时间内能进行一次服务过程，并且用户的个人信息不被泄漏或随意串改，所以要求系统具有较高的稳定性和安全性，降低系统出错率，才能保证系统正常、高效、快速的使用，所以本系统在性能上还需要具备以下关键指标：*

1. 系统应支持至少1万个司机用户用户和1万个打车乘客用户使用。
2. 系统应支持1万个用户同时申请打车服务。
3. 系统应支持200个停车场管理员同时上传、修改、上传停车场相关信息，且该情况下系统的响应时间不得超过5秒。
4. 系统应支持1万个用户同时并发登陆系统，且系统产生的延迟不超过5秒钟。
5. 系统将信息更新至数据库时，对数据的处理量在G级别时，系统的响应时间应该在5秒之内。
6. 在系统因并发访问人数过多而发生宕机时，系统应该在10分钟内能进行系统恢复。
7. 对处理数据量的要求：

对数据的处理量在G级别时，系统的响应时间应当在 5s 之内。

所以本系统在可靠性上应当：

1. 出错率

在累计 100000 次服务请求的情况下，其出错率不得高于 3 次。

在累计 1000000 次服务请求的情况下，其出错率不得高于 60 次。

1. 稳定性

在 10000 用户的并发操作下，连续 5 个小时工作的情况下，系统不会出现停止服务的情况。

1. 易恢复性

在停机进行恢复之后损失的数据不得大于1 h。

## 2.2体系结构用例

出行乘客：有出行需求、需要打车的用户。

司机用户：持有合法驾照，能够提供开车送乘客去目的地服务的驾驶员。注意：为了司机用户的行驶安全，要求本打车软件对语音输入友好，并且界面简单，功能入口明显。

在线客服：帮助用户解决软件使用中遇到的问题的系统管理人员，能协调出行乘客与司机用户的系统管理人员。

软件的监测人员：要能获取到打车请求信息，并能顺利传递。

测试人员：明确系统的性能指标。 明确系统的组织结构。 编写测试用例的依据

## 2.3各相关方对体系结构的要求

打车软件建立了如上图的“4+1”体系结构视角模型，该模型从4个角度（逻辑、实现、进程和部署）指出不同的相关利益方关心的事情，外加从使用者的角度对用例做观察分析，分析其影响系统的上下文和商业目标情况。

最终用户、设计者要求系统可以实现需求分析阶段的基本需求。要求本系统提供给出行乘客用户、司机用户用户、客服进行使用。通过限制不同角色的使用权限，从而达到出行乘客用户申请打车服务，司机用户乘客提供并实现服务的目标，在节省了人力资源的同时不失安全性和可靠性。

用户、分析人员和测试人员要求系统在最终使用的时候可以可靠完整的运行，数据展示得当，带给用户良好的视觉体验性。

系统集成人员主要着重于系统的性能方面。比如系统时间性能和空间系能上的要求，这些要求保证了系统的未来的可增长性。

系统工程师主要关注于系统的组织结构、交付时间、系统设计时和用户的交流以及在系统完成时对停车场负责人进行相应的培训，以便其正确的使用系统。

## 2.4约束条件

|  |  |
| --- | --- |
| **约束** | **描述** |
| **项目进度要求** | 项目工期约束，即开发时间为30天 |
| **系统开发成本** | 10000～15000元 |
| **项目开发要求** | 使用Java EE编写，便于维护 |
| **业务约束** | 用户可使用本系统成功预约车位 |
| **审查机制** | 在软件开放过程中，每隔两周需要对软件开发方进行定期工期检查。 |
| **性能** | 正常情况响应时间小于2秒，最大并发量下响应时间控制在5秒以内 |
| **资源管理** | 服务器部件的内存必须具有40%以上的预留量 |
| **可使用性** | 系统需要保证24x365正常运行，可使用性达到99%以上 |
| **易用性** | 系统是web应用，用户无需安装客户端 |
| **可靠性** | 不允许丢失信息 |
| **密安性** | 所有的修改请求必须被授权，并使用认证过程进行加密 |

## 2.5非功能需求

### 2.5.1 可靠性

* 本系统测试时间至少为100小时，测试参与用户至少为200人。本系统应该避免因为故障而导致的失效的能力，在进行测试的过程中，本系统应该尽可能遍历所有的可能故障情况，保证系统的成熟。
* 本系统在发生硬件或者软件异常时，应该具有依然具有服务能力。服务能力虽然下降，但不会达到系统崩溃的程度级别。
* 本系统在发生停机故障之后，应该保证十分钟之内可以修复并且正常运行。

### 2.5.2 可用性

* 在接收了用户错误的数据输入之后，本系统能不死机、不崩溃，依然正常工作。本系统应该可以进行错误识别并且给出用户相应的提示信息。
* 系统界面美观，吸引使用者。本系统的界面设计应该在保证基本功能的基础上做到布局美观友好，系统界面和市场面上大部分管理系统界面类似，用户的体验性在本系统的界面设计上因该充分提高。
* 本系统易于学习。本系统应该保证新用户在初次使用时，在不超过3分钟的学习时间之内便可以明白整个系统的基本功能，在10分钟的学习时间之内该用户应该熟练掌握系统的全部功能。
* 用户使用本系统工作效率得到提高。本系统应该节约了用户查找空车位的时间，并且本系统应该提高了停车管理人员的管理停车场效率。
* 系统提供相应的操作指南文档。本系统提应该提供给使用者相应的文档使用手册，便于使用者获得系统功能的详细解读和使用。

### 2.5.3 安全性

* 本系统应该对使用者的权限进行控制。本应同应该使用手机号、姓名、身份证号进行实名注册，并使用手机号和密码进行系统的登录，不同的类型的使用者应该具有不同的系统使用权限。本系统通关系统管理员管理实现对于不同角色开放不同功能的行为，来防止不合法的操作的产生。
* 本系统对于重要数据应该进行数据加密。初次登录系统时，应该输入相应的用户名和密码，对于密码的存储应该采取一定的加密措施，防止因为管理员不适当的信息泄露而产生不可挽回的错误。
* 系统数据定期备份。为了本城市共享停车管理系统的长远考虑，系统数据应该进行相应的备份，防止在系统停止工作时能保证进行数据恢复。

### 2.5.4 可维护性

* 在系统运行的过程之中，系统容易诊断出缺陷和失效问题，并且给出行相应的运行时提示。
* 本系统的的代码和文档结构同样应该保证清晰易读，便于在系统主线错误之后进行快速的回复修改，能准确的找出系统错误的原因。
* 在进行系统升级的时候，应该进行版本的控制，减少不必要的功能升级，防止因为升级修改次数越多系统越不稳定现象的发生。
* 本系统耦合度低，便于完成后期系统测试。

### 2.5.5 可移植性

* 本系统应该保证在市面上常用的系统环境下便于进行安装使用，不会因为环境的不同而导致系统的崩溃或者时系统功能的缺失。
* 本系统的接口设计合理，便于进行系统的改造和重用，应该方便被当作模板框架来进行相应其他系统的二次开发。
* 本系统在和其他系统在同一个平台下面进行运行的时候，不会产生冲突，各自有条不紊的运行。

### 2.5.6 性能

* 系统的一般操作相应时间应该不超过2秒，系统涉及数据连接的操作相应时间应该不超过5秒。
* 在有300个以内用户同时登录进行成绩查询时，本系统应该不会有超过6秒的信息反馈延迟；当用户数量超过500人时，本系统应该不会有超过8秒的信息反馈延迟。
* 在系统因为访问人数过多而发生宕机时，本系统应该在10分钟之内可以进行恢复。
* 本系统应该存储至少2万人的信息的记录，用于不同身份背景的人进行系统的使用。
* 本系统的磁盘容量应该至少为10TB，其中应该有至少300GB的空闲空间，用于信息的扩充。

## 2.6风险

1. 对需求变化的适应性：在软件的开发过程中，用户很可能会中途更换需求，或者要求增加新的需求，体系结构文档能否适应新的需求是一个大问题。
2. 工期的综合要求：用户通常会对工期提出要求，比如第一个版本在6个月内完工，第二个版本在9个月内完工，第三个版本要在12个月内完工等等。体系结构的设计能否很好的支持上述要求。
3. 软件易用性：软件的易用性是影响软件是否被用户接受的关键因素。在软件产品中，设计复杂，功能强大而完备，但因为操作繁复而被搁置者屡见不鲜。造成的主要原因在于缺乏软件开发中软件体系结构的宏观把握能力。
4. 软件的可伸缩性：是指软件在不进行修改的情况下适应不同的工作环境的能力。由于硬件的飞速发展和软件开发周期较长的矛盾，软件升级的需要显得非常迫切。如果软件的升级和移植非常困难，软件的生命期必定很短，使得化费巨大人力物力开发出的软件系统只能在低性能的硬件或网络上运行，甚至被废弃不用，造成巨大的浪费。
5. 软件的可维护性：软件的维护也是必然的事情，为了保证软件的较长使用寿命，软件就必须适应不断的业务需求变化，根据业务需求的变化对软件进行修改。修改的成本和周期都直接和软件的体系结构相关。一个好的软件体系结构可以尽可能地将系统的变化放在系统的配置上，即软件代码无需修改，仅仅是在系统提供的配置文件中进行适当的修改，然后软件重新加载进入运行状态，就完成了系统部分功能和性能要求的变化。对于重大改动，需要打开源代码进行修改的，也仅仅是先继承原先的代码，然后用新的功能接替原先的调用接口，这样将把软件改动量减小到最低。

# 3.解决方案

## 3.1相关体系结构

### 3.1.1B/S体系架构

根据调查统计，当前市面上大部分相关的系统都是在抽象意义上采用B/S体系结构模式，在具体编程实现时采用MVC体系结构模式。

B/S体系结构的优点就是可以在任何地方进行操作而不用安装任何专门的软件。只要有一台能上网的设备就能使用，客户端零维护。系统的扩展非常容易，只要能上网，再由系统管理员分配一个用户名和密码，就可以使用了。甚至可以在线申请，通过公司内部的安全认证（如CA证书）后，不需要人的参与，系统可以自动分配给用户一个账号进入系统。

MVC系统体系结构有以下优点：在目前用户需求的快速变化下，可能有多种方式访问应用的要求。减少了代码的复制，即减少了代码的维护量，一旦模型发生改变，也易于维护。 其次，由于模型返回的数据不带任何显示格式，因而这些模型也可直接应用于接口的使用。

再次，由于一个应用被分离为三层，因此有时改变其中的一层就能满足应用的改变。一个应用的业务流程或者业务规则的改变只需改动MVC的模型层。 控制层的概念也很有效，由于它把不同的模型和不同的视图组合在一起完成不同的请求，因此，控制层可以说是包含了用户请求权限的概念。最后，它还有利于软件工程化管理。由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化产生管理程序代码。

### 3.2体系结构概述

本系统结合B/S体系结构模式和MVC体系结构模式设计而成。

B/S架构主要体现在使用者借助浏览器登录软件管理系统，开始进行相关数据的反馈，根据客户相关的业务逻辑，数据将会从数据库中取出同样显示在网页上面。MVC主要用于编写代码时的约束，使得设计思路清晰，便于未来系统的修改和重新使用。

本系统采用B/S架构的原因如下：

（1）B/S 最大的优点就是可以在任何地方进行操作而不用安装任何专门的软件。只要有一台能上网的设备就能使用，客户端零维护。系统的扩展非常容易，只要能上网，由系统管理员分配一个用户名和密码就可以使用。甚至可以在线申请，通过公司内部的安全认证（如CA证书）后，不需要人的参与，系统可以自动分配给用户一个账号进入系统。B/S这种优点非常符合本系统的要求，因此采用B/S架构进行系统的体系构建。

（2）B/S架构建立在广域网之上的，不必是专门的网络硬件环境。例与电话上网，租用设备。信息自己管理，比C/S更强的适应范围，一般只要有操作系统和浏览器就行。因此本系统客户端采用浏览器进行访问的时候不用考虑网络因素的限制，保证了信息的反馈速度。

（3）系统维护方便，构件组成方便构件个别的更换，实现系统的无缝升级。系统维护开销减到最小，用户从网上自己下载安装就可以实现升级。

（4）建立在浏览器上，有更加丰富和生动的表现方式与用户交流，并且大部分难度减低,减低开发成本。

（5）对安全以及访问速度的多重的考虑，建立在需要更加优化的基础之上。比C/S有更高的要求，B/S结构的程序架构是发展的趋势，全面支持网络的构件搭建的系统，SUN 和IBM推的JavaBean 构件技术等使 B/S更加成熟。

### 3.3结构化视图

### 3.3.1概念级体系结构

MVC体系架构主要体现在打车软件，一共分为三层：表示层、应用层、服务层。这三层最终都是基于B/S架构的浏览器和服务器实现的。这三层主要用于编写代码时的约束，使得设计思路清晰，便于未来系统的修改和重新使用。

1.表示层

表示层位于用户客户端，该层的相关功能是接受用户从客户端浏览器发送的业务请 求，将相关请求发送到逻辑业务处理层，并将逻辑业务处理层反馈的处理结果输出到浏览 器上，供用户浏览。B/S 体系架构中的客户端使用人们常用的 Web 浏览器作为载体，输 入系统服务器所在网址，自动弹出下载插件的通知，将其装在浏览器中即可供用户使用， 无需下载安装程序，减少了客户端应用程序安装、运营更新的需求，大大的降低了网络管 理员的工作量。

2.逻辑业务处理层(Web 服务器层)

逻辑业务处理层位于 B/S 体系架构中的 Web 服务器上，也被很多场合成为 Web 服务 器层。逻辑业务处理层的相关功能是完成系统客户端发送过来的用户请求，由 Web 服务器分配处理任务，执行用户请求的逻辑业务，同时并根据用户的需求检测其是否需要数据 处理，如果需要数据处理，则将处理结果的请求发送到数据库服务器上，由数据库服务器 完成余下工作，同时将数据库服务器处理的结果反馈到客户端，供用户浏览。

3. 数据层(数据库服务器层)

数据层位于数据库服务器上，通常又被称为数据库服务器层，其相关功能是处理逻辑业务处理层发送过来的数据业务请求，该层可以使用数据库管理系统的相关操作，实现数

据请求的添加、删除、修改等操作，同时将数据处理结果及时的反馈到逻辑业务处理层。

MVC 和 B/S 组织模型

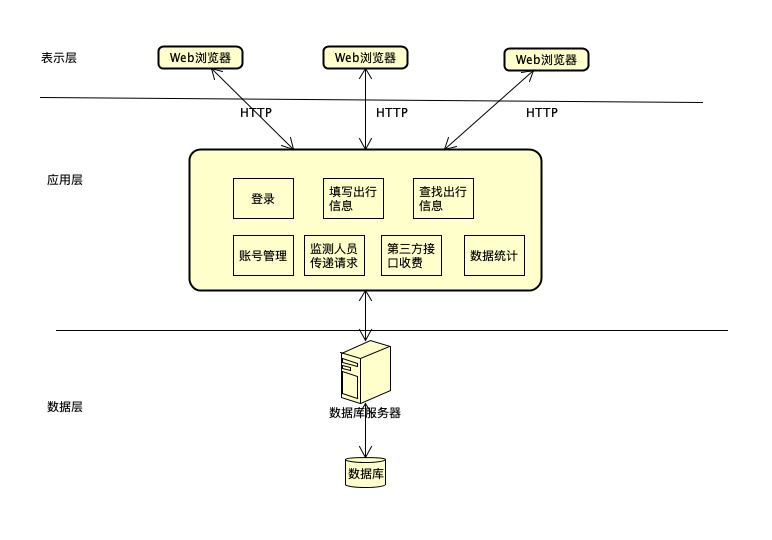
|  |  |  |
| --- | --- | --- |
| MVC层次 | 理解性抽象 | 对应B/S架构 |
| 表示层（V） | 用户界面层 | 浏览器获取数据 |
| 应用层（C） | 用户身份验证层、信息检索和修改层 | 浏览器显示数据，服务器传送数据 |
| 服务层（M） | 数据库层 | 服务器计算数据 |

用户界面层是和用户最直接接触的一个层次。这个层次通过网页界面进行相应的展示，用户可以通过自己的浏览器界面进行相应的访问和查看。

用户身份验证层是进行用户身份类型的检查。本系统面向出行乘客用户、司机用户用户、在线客服等进行打车软件的管理和使用，对于不同身份的人系统提供的功能是不同的。在实际中，采用识别出相应的用户ID来完成身份的验证，因此用户身份验证层是不可缺少的。

信息检索和修改层主要针对的是空闲车辆查询、出行乘客用户申请乘车以及司机用户用户的提供服务、完成服务后与第三方接口实现。在这一层根据上一层获得的身份类型返回不同的查询结果，比如普通用户（司机用户）只能查询到司机用户本人的余额信息以及查询到自己车辆信息。停车管理员只能修改其所管辖的停车场及设备信息。

数据库层主要完成数据信息的存储和备份，根据上一层获取的条件返回相应的查询结果，并且对数据进行相应的权限管理，防止非管理人员进行相应的数据访问。



### 打车软件概念级视图

### 3.3.2模块级体系结构

系统的模块化设计是系统进行复用的关键。模块级体系结构反映了对软件代码实现时 的期望，特别是对于程序规模较大的系统。

模块化表达有两种方式：

1）将系统按功能从逻辑上分解为系统、模块以及程序单元；

2）按系统的层次进行划分。

本说明书根据《打车软件系统需求规格说明书》将系统按功能从逻辑上分解，细化功能结构模块。

#### 3.3.2.1 出行乘客用户交互模块

出行乘客用户交互模块是针对出行乘客用户用户设计的。主要设计思想如下：

(1) 系统要求出行乘客用户用户在进入系统之前输入用户名和密码进行信息的验证。

(2) 系统要求当密码校验失败时，会提示错误，如果超过系统所允许的最大连续出错值，那么会对该用户进行锁定。

(3)系统应该能够实现让出行乘客用户可以根据自己的喜好来修改自己的用户名称和密码的功能。

(4)为了建立完善可靠的互联网信用基础，应该让出行乘客用户进行实名认证身份证号码 。

(5)为了实现自动扣费，出行乘客用户应绑定至少一个支付手段。

(6)系统应该能够实现让出行乘客用户发布订单时设置多种条件，如出发时间，上车地址，目的地址，车辆类型等。

#### 3.3.2.2司机用户用户交互模块

司机用户交互模块是针对司机用户用户设计的。主要设计思想如下：

(1) 系统要求司机用户用户在进入系统之前输入用户名和密码进行信息的验证。

(2) 系统要求当密码校验失败时，会提示错误，如果超过系统所允许的最大连续出错值，那么会对该用户进行锁定。

(3)系统应该能够实现让司机用户可以根据自己的喜好来修改自己的用户名称和密码的功能。

(4)为了建立完善可靠的互联网信用基础，应该让司机用户进行实名认证身份证号码和车牌号验证，并与交通管理部门取得联系。

(5)为了实现自动扣费，使得线下停车朝着数据化、无人化与智能化方向发展，司机用户应绑定至少一个支付手段。

(6)系统应该能够实现让司机用户可以通过多种策略挑选订单，如就近查看，抢单，红包等等。

(7)为了实现快速到达目的地址，智能分析系统应该根据路况规划多种路线，由司机用户选择最佳路线。

#### 3.3.2.3系统检测管理员交互模块

业务监测员交互模块是针对业务监测员用户设计的。主要设计思想如下：

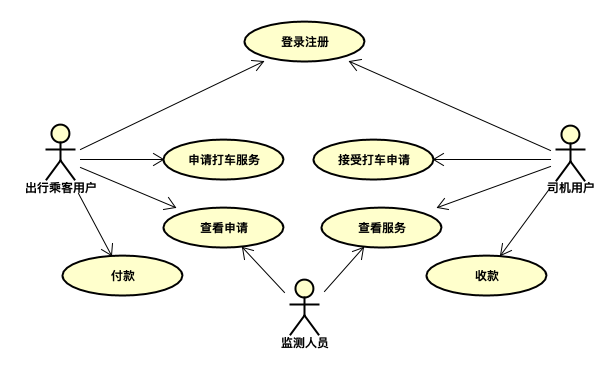
(1) 系统要求业务监测员用户在进入系统之前输入用户名和密码进行信息的验证。

(2) 系统要求当密码校验失败时，会提示错误，如果超过系统允许的最大连续出错值，那么会对该用户进行锁定。

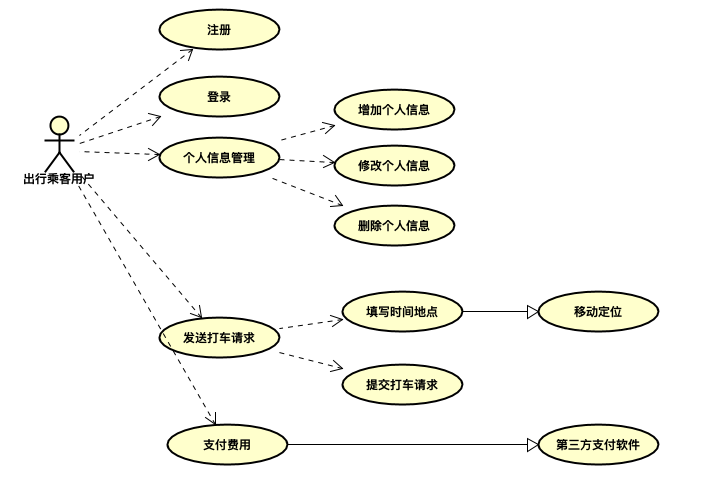
(3) 系统应该能够实现对用户提交申请的订单的出行乘客用户和司机用户的信息查看的功能，修改出行乘客用户和司机用户的功能权限，发布通知。

# 4.用例视图

用例视图，又称“场景视图”，它综合所有的视图。用于刻画构件之间的相互关系，描述了代表一些重要的核心功能的场景或用例集。可以描述一个特定的视图内的构件关系，也可以描述不同视图间的构件关系。



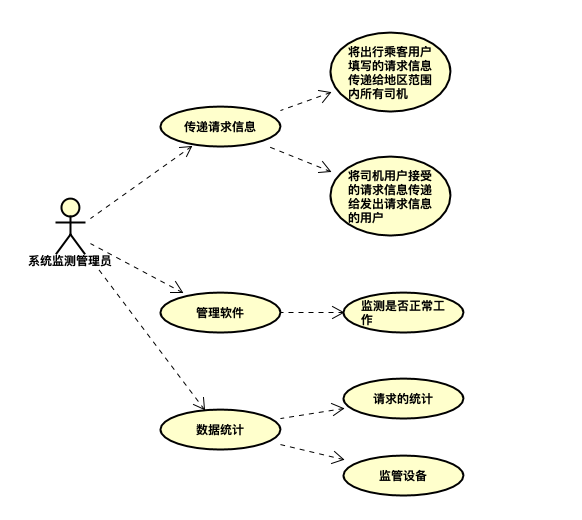
## 4.1出行乘车用户



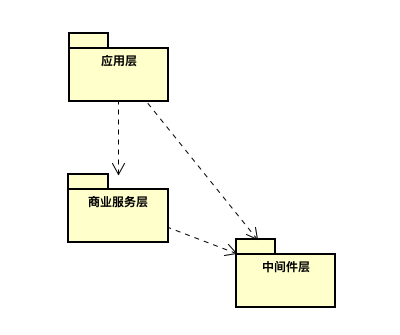
## 4.2司机用户

## 屏幕快照%202019-11-18%2014

## 4.3系统监测管理员



# 5.逻辑视图



逻辑视图描述了重要的类和用例实现，说明了结构上重要的类、子系统、包和层之间的关系。

5.1.1应用层

该应用程序层具有代表用户看到的应用程序屏幕的所有边界类。该层取决于过程对象层；跨越了客户与中间层的分离。

5.1.2商业服务层

商业服务层具有所有控制器类，这些控制器类代表驱动应用程序行为的用例管理器。该层表示客户端到中间层的边界。业务服务层取决于流程对象层；跨越了客户与中间层的分离。

5.1.3中间件层

中间件层支持对关系DBMS和OODBMS的访问。

# 6.进程视图

进程视图是对体系结构的过程视图的描述。描述系统执行中涉及的任务（进程和线程），它们的交互作用和配置。还描述了对象和类对任务的分配。

## 6.1进程视图

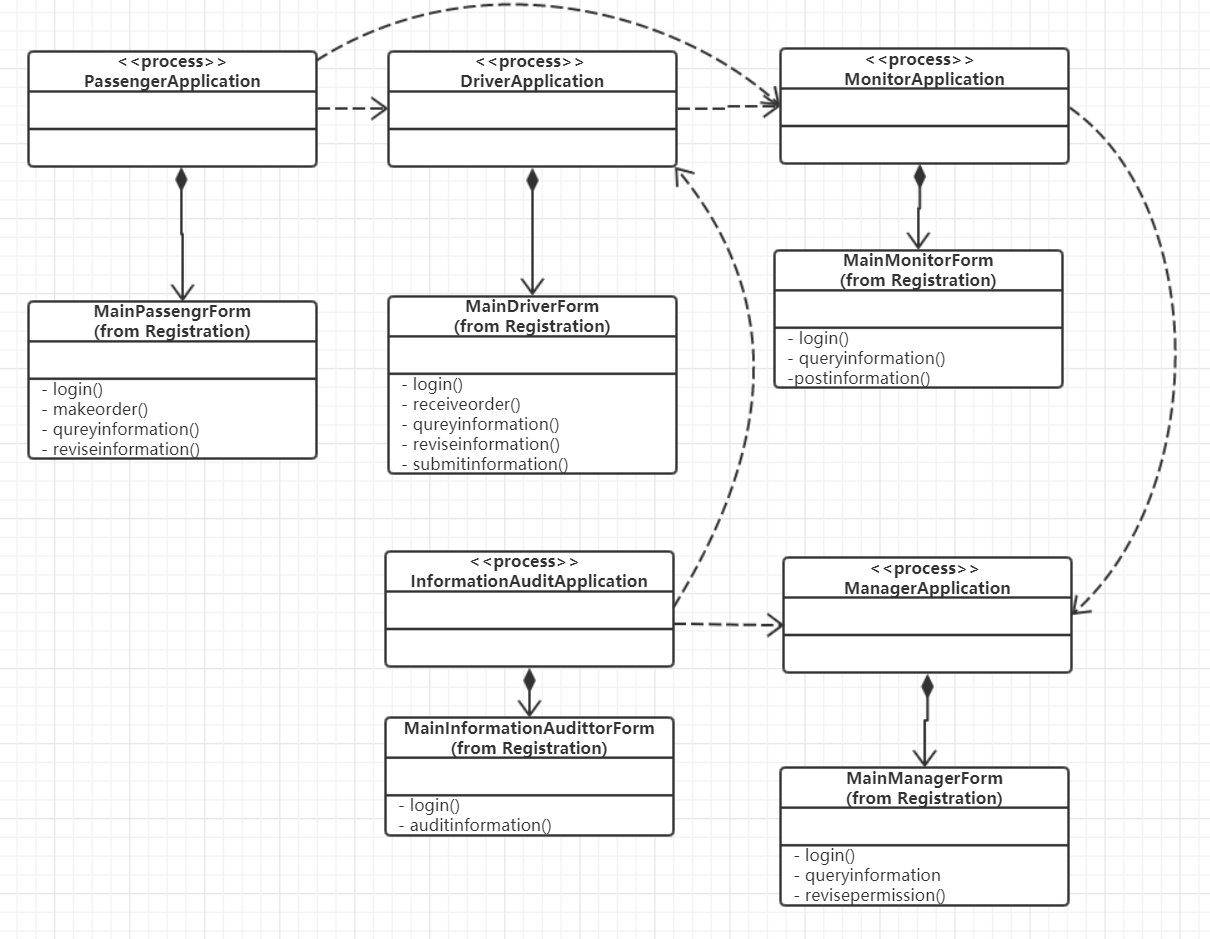


图6-1系统进程视图

### 6.1.1 出行乘客用户发布订单

出行乘客用户根据个人要求发布订单，发布订单时可以设置多种条件，如出发时间，上车地址，目的地址，车辆类型等。

### 6.1.2 出行乘客用户查询订单

出行乘客用户可以查询自己的历史订单和当前订单的详细信息，包括出发时间，上车地址，目的地址，乘坐车辆的司机用户，车辆类型，司机用户的联系方式，费用等。

### 6.1.3 出行乘客用户反馈问题

出行乘客用户在订单结束后，可以向系统检测管理员反馈订单中出现的问题，由系统检测管理员负责协商解决。

### 6.1.4 司机用户接单

司机用户可以通过多种策略在软件系统中挑选订单，如就近查看，抢单，红包等。

### 6.1.5 司机用户查询订单

司机用户可以查询自己的接单记录和当前订单的详细信息，包括出发时间，上车地址，目的地址，乘客的联系方式，费用等。

### 6.1.6 司机用户反馈问题

司机用户在订单结束后，可以向系统检测管理员反馈订单中出现的问题，由系统检测管理员负责协商解决。

### 6.1.7 司机用户提交信息

司机用户可以向平台提交相关信息，由车辆信息审核人员负责审核，如果审定无误，系统应当实现让车辆信息审核员将司机用户提交的信息录入数据库，并赋予司机用户相应的权限，提供给司机用户相对应的服务。

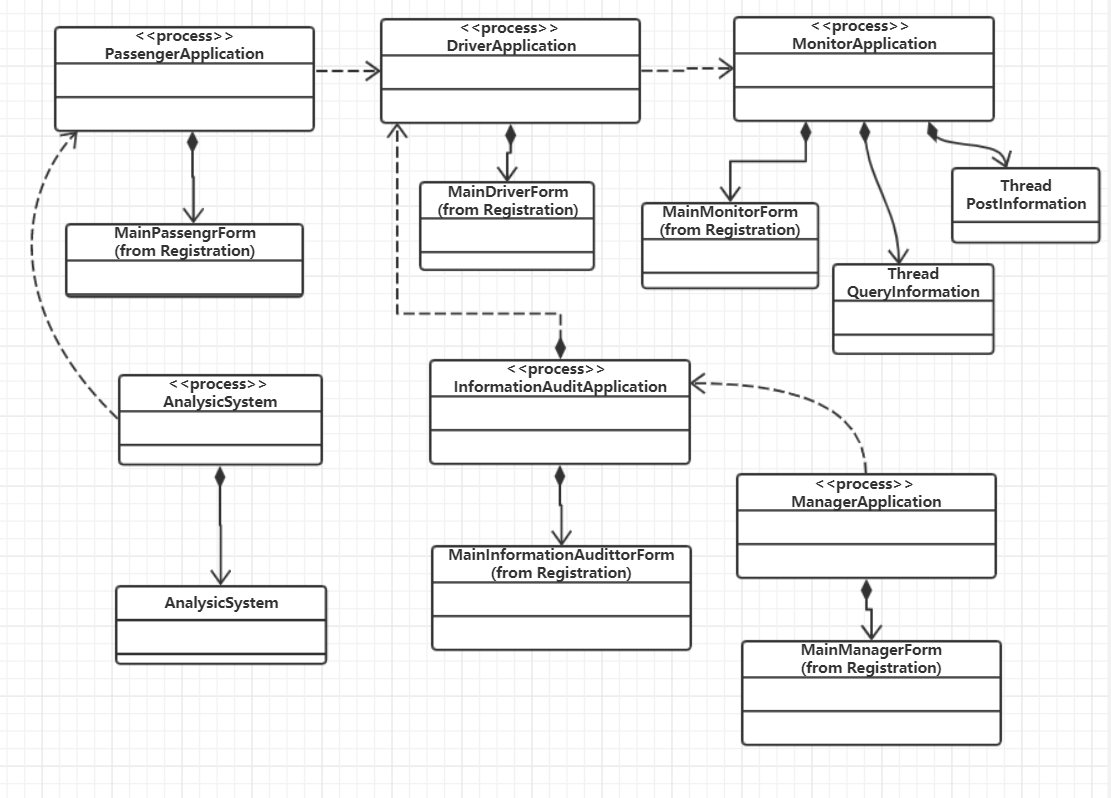
### 6.1.8 系统检测管理员查询信息

系统检测管理员可以对用户提交申请的订单进行信息查询。

### 6.1.9 系统检测管理员发布信息

系统检测管理员可以对出行乘客用户用户和司机用户用户发布信息，对订单中发生的问题进行处理解决。

## 6.2设计元素的过程



进程元素设计图

### 6.2.1发布订单

出行乘客用户可以发布订单，订单信息包括上车地址，目的地址，选择的车辆类型，相应的预计价格。

### 6.2.2 接单

司机用户可以接单，根据系统通知，接收附近的打车信息，确认订单。

### 6.2.3 查询订单

出行乘客用户和司机用户可以在个人订单中查看历史订单和当前订单的详细信息。

### 6.2.4 发布信息

系统检测管理员可以向出行乘客用户和司机用户发布信息。

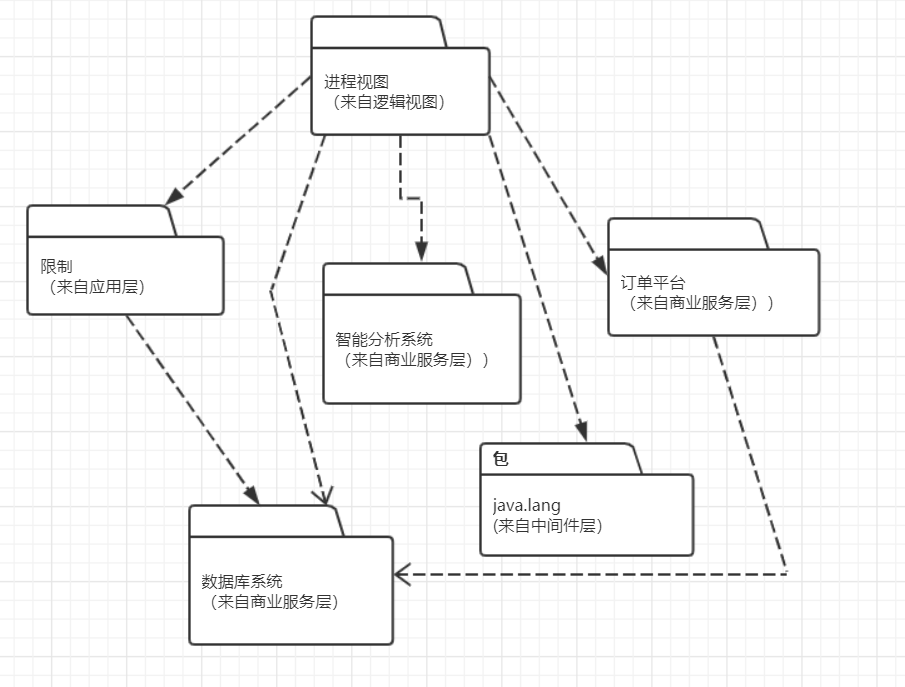
### 6.2.5 审核信息

车辆信息审核人员可以审核司机用户提交的相关信息。

### 6.2.6 修改权限

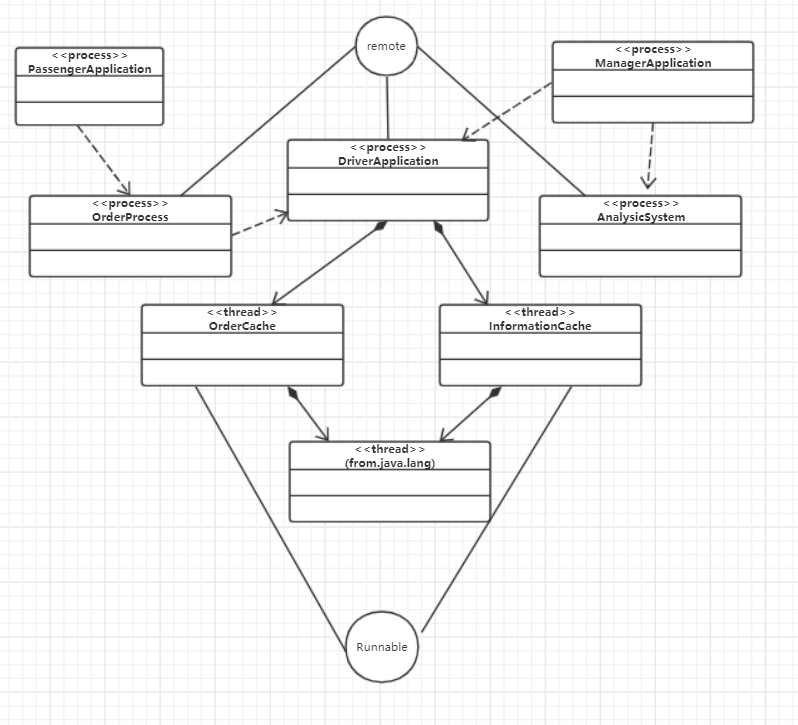
系统管理人员可以修改用户的权限。

## 6.3过程模型设计模型的依赖关系



设计模型相关性的过程模型

## 6.4实施过程



实施过程

### 6.4.1远程

远程接口用于识别所有远程对象。任何作为远程对象的对象都必须直接或间接实现此接口。远程接口中仅指定了那些方法。实现类可以实现任意数量的远程接口，并且可以扩展其他远程实现类。

### 6.4.2可运行

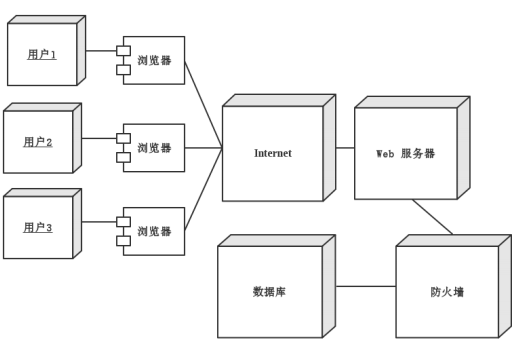
可运行接口应该由实例旨在由线程执行的任何类实现。该类必须定义一个没有参数的方法，称为run。此接口旨在为希望在活动状态下执行代码的对象提供通用协议。例如，Runnable由Thread类实现。处于活动状态仅表示线程已启动但尚未停止。

### 6.4.3线程

线程是程序中的执行线程。 Java虚拟机允许应用程序具有多个并发运行的执行线程。每个线程都有一个优先级。具有较高优先级的线程优先于具有较低优先级的线程执行。每个线程可能会也可能不会被标记为守护程序。当在某个线程中运行的代码创建新的Thread对象时，新线程的优先级最初设置为与创建线程的优先级相等，并且当且仅当创建线程是守护程序时，该线程才是守护程序线程。

# 7.部署视图

从系统软硬件物理配置的角度，描述系统的网络逻辑拓扑结构。模型包括各个物理节点的硬件与软件配置，网络的逻辑拓扑结构，节点间的交互和讯关系等。同时还表达了进程视图中的各个进程具体分配到物理节点的映射关系。由于本系统采用的是 B/S 架构，结 合现实的网络拓扑结构



系统部署视图

## 7.1手机

出行乘客用户和司机用户提供手机使用该软件。

## 7.2计算机

系统检测管理员、车辆信息审核人员和系统管理人员通过计算机执行业务。

## 7.3 注册服务器

该软件被部署在大型服务器上，由专业人员进行维护。

## 7.4 数据库

数据库存储该软件相关的所有数据，由专业人员负责数据库安全。

## 7.5 智能分析系统

智能分析系统负责该软件的业务处理。

# 8.性能

## 8.1时间性能：

(1)系统执行速度

系统执行速度需要满足以下条件：系统响应快。数据库单表操作时间不大于0.5 秒。

(2)系统响应时间

考虑到同时进行操作的用户人数，系统的响应时间应该达到如下标准比较合理：

* 系统应当可以查看和修改用户的基本信息。一个流程的响应速度最佳为 b1

s,平均为 a1 s，不超过 m1 s。系统应当可以同时接受 10000 人同时操作。

## 8.2可靠性

系统能较长时间下稳定运行，同时该系统需要具备一定的故障恢复能力，即有 一定的容错能力。当用户的操作不当引起某些故障时，或者是由于操作系统或者网络发生故障时，系统需要具备一定的故障恢复能力。选择数据库产品时，要考虑一定的数据负载能力。由于在处理员工信息、客户信息、账户信息等信息时，系统需要做大量的数据统计和处理，因此要具备相应的数据负载能力。

## 8.3易用性

* 该系统的可使用性体现在它可以支持多操作系统多浏览器运行。同时该系统应该具备容易操作的功能。

（1）提供针对不同用户的用户使用说明手册，方便用户学习使用。

（2）系统应提供在线帮助界面，方便用户学习操作。

* 由于操作该系统的人员有很多，且操作习惯、受教育程度、年龄阶段、接受事物能力等都各不相同，这就要求系统具备良好的人机交互能力。系统提供的各种功能便于用户理解，操作简单，用户很容易掌握。

(1)系统的界面设计应简洁明了，使用户能够自己学会使用本系统。

(2)系统应具有一定的美观性，可参考目前大部分网站和手机应用程序的扁平化设计。

## 8.4密安性

（1)通常来讲，实际使用的管理系统，必须具备相应的安全性能。该系统各级用户有各自的权限设置，例如用户之间不可以互相查看或修改其他人的个人信息。

（2）该系统应该通过设置防火墙确保数据传输的安全。使用可靠的操作系统来保证系统的操作安全，确保系统在一个安全可靠的环境下运行。

（3）系统应保证用户信息不泄露，系统配置文件和数据库存储文件应当进行加密处理。

（4）系统应保证不会因恶意攻击而崩溃，系统开发过程不存在明显漏洞。

（5）系统应当能够保证选取的开发方不存在商业竞争对手或类似的恶意对手。

## 8.5可维护性

软件的可维护性是指改进软件的难易程度。该系统的结构、接口、功能以及内部过程在开发以及跟踪阶段，容易被维护人员理解。同时，该系统有良好的测试和诊断系统错误的功能。当系统应用于不同城市的公共交通系统之下时，应该具备良好的适应性。不需要通过大幅度的接口与内部过程修改，就能使用户进行使用。

## 8.6可移植性

（1）易安装性：该系统能够跨平台移动运行，包括 Windows 服务器平台以及 Linux 平台，和IOS系统以及安卓系统。

（2）共存性：系统应当能够和其他软件共存于一个平台上，存在冲突的软件不超过。

（3）已替换性：系统可被容易地卸载，也易被高版本的系统替换。

## 8.7可扩展性

为了满足不管发展的客户需求和业务需求，系统安装后，在后续的功能维护 和拓展中，需要具备良好的维护性及可拓展后，便于日后系统的升级和修改。

# 9.系统的质量分析和评价

除系统功能性需求外，系统的非功能性需求也越来越重要，一个软件产品的性能与非功能性需求有很大关系。非功能性需求往往是针对一个子系统或模块进行的，而不能与每一个功能需求条款一一对应。主要包括质量要求、工程要求和其他要求。质量要求又包括性能，可靠性，可用性，安全性，可维护性，可移植性等；工程要求包括反要求，设计约束和逻辑数据库要求。

## 9.1场景分析

系统的使用场景与质量属性是密切相关的，也是决定体系结构的重要依据。通过场景 可以很好的评估体系结构，本文使用 SEI 开发的体系结构折中分析方法

### 9.1.1用例场景

用例场景是从使用者角度出发，描述用户所期望的与这个系统的交互。下面是用例场景的几个例子。

**场景1**：**出行乘客用户**希望可以发布订单，当自己的订单被接收后，可以查看接单的司机用户的车辆类型和联系方式以及司机用户的当前位置。该场景表明了用户期望系统易于使用，即易用性。

**场景2：司机用户**希望可以查看到自己的附近的用户发布的订单，接单后系统可以规划从当前位置到上车地点的路线。该场景表明了用户期望系统易于使用，即易用性。

**场景3：系统检测管理员**希望可以通过系统查询订单信息和用户信息，并向用户发送信息。该场景表明了用户期望系统易于使用，即易用性。

**场景4：**当系统检测管理员接到用户的订单反馈时，可以及时对订单问题进行处理。该场景表明了用户期望系统易于使用，即易用性。

**场景5：**用户对订单提出问题，期望可以在12h内获得结果。该场景表明了系统的性能需求。

**场景6**：当用户改变路线后，系统要在1s内更新。该场景表明了系统的性能需求。

**场景7：**当处理器发生故障后，缓存系统要能在1s中内从一个处理器切换到另一个处理器，该场景代表了系统可靠性要求。

### 9.1.2增长性场景

**增长性场景1：**未来可能会需要更改用户的UI界面，这只需要更改表示层的逻辑即可，而不需要更改业务逻辑层和数据层的代码，希望系统仅需增加2人周的工作量就能完成修改。

**增长性场景2：**未来可能会增加新的数据服务器，这将会降低用户的Web访问时间，希望系统仅需增加1人周的工作量就能完成修改。

**增长性场景3：**通过扩充现有数据库表的规模和在特定的列上创建索引，可以降低平均检索时间到1s以内。

### 9.1.3探索性场景

**探索性场景1：**系统可以和交通管理系统对接，对交通状况进行实时更新，并根据实时交通状况规划路线。不超过5个月的工作量。

**探索性场景2：**正常情况下，当服务器有一半宕机时，不影响整个系统的可用性。

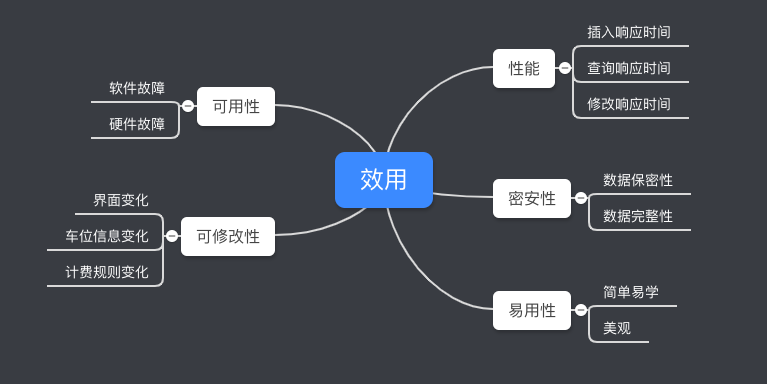
**探索性场景3：**当系统的数据库出现问题时，可以通过日志文件进行恢复，损失的数据不超过5分钟。

## 9.2原型分析

### 9.2.1原型评价法

1. 写一个测试程序，作为司机用户用户，在不断地调用查询条件30000次，看其需要多长时间。
2. 写一个测试程序，作为管理员用户（不同类别的管理员均包含在内），在不断地调用查询插入语句3000次，看其需要多长时间。

### 9.2.2效用树法



效用树图

## 9.3风险

1. 项目进度要求：产品的第一个版本必须在6个月内完成，而且第二个版本要在九个月内完成，第三个版本要在12个月内完成。本体系结构是否可以很好的支持进度要求。
2. 项目需求变化：在开发过程中用户经常会要求改变需求或者增加需求，此时，体系结构是否还能适用。

3）软件易用性：软件的易用性是影响软件是否被用户接受的关键因素。在软件产品中，设计复杂，功能强大而完备，但因为操作繁复而被搁置者屡见不鲜。造成的主要原因在于缺乏软件开发中软件体系结构的宏观把握能力。

4）软件的可伸缩性：是指软件在不进行修改的情况下适应不同的工作环境的能力。由于硬件的飞速发展和软件开发周期较长的矛盾，软件升级的需要显得非常迫切。如果软件的升级和移植非常困难，软件的生命期必定很短，使得化费巨大人力物力开发出的软件系统只能在低性能的硬件或网络上运行，甚至被废弃不用，造成巨大的浪费。

5）软件的可维护性：软件的维护也是必然的事情，为了保证软件的较长使用寿命，软件就必须适应不断的业务需求变化，根据业务需求的变化对软件进行修改。修改的成本和周期都直接和软件的体系结构相关。一个好的软件体系结构可以尽可能地将系统的变化放在系统的配置上，即软件代码无需修改，仅仅是在系统提供的配置文件中进行适当的修改，然后软件重新加载进入运行状态，就完成了系统部分功能和性能要求的变化。对于重大改动，需要打开源代码进行修改的，也仅仅是先继承原先的代码，然后用新的功能接替原先的调用接口，这样将把软件改动量减小到最低。

# A．附录