<顺风打车系统>

体系结构设计文档

<3.0>

<2019.12.14>

吴志镛

2017211869
2017211501 班

软件工程导论
2019 秋

修订记录

日期	描述	作者	内容
<2019-12-1>	<版本 1>	<吴志镛>	<第一次修订>
<2019-12-13>	<版本 2>	<吴志镛>	<第二次修订>
<2019-12-14>	<版本 3>	<吴志镛>	<第三次修订>

文档审批

以下的软件需求规格说明书已被接受并认可：

签名	打印版姓名	职位	日期
	<吴志镛>		

目录

1.介绍

在这一部分，主要介绍了本体系结构文档编写的目的、范围、相关概念定义、以及所使用的参考文献。软件体系结构设计属于高层设计文档，是符合现代软件工程要求的概要设计。

1.1 目的

本文档为体系结构设计文档。旨在从设计的角度对顺风打车的总体结构（逻辑设计、物理结构等）进行综合的描述，并给出了该软件系统设计的总体策略和相应技术解决方案，对软件系统的质量以及开发过程的风险进行了评估。方便了开发过程中不同人员间的交流，提高了软件系统开发的效率。

用户可通过本文档了解软件系统的总体功能和整体架构；开发人员可以通过本文档了解该软件系统预期的功能以及相应的体系架构，并以此为指导对系统进行设计和开发；测试人员可以通过此文档选择与该系统体系结构相适应的测试模型与流程，针对性地进行系统测试；维护人员可根据本文档知道如何对本系统进行维护并在实际应用场景下，进行相关参数的调整；项目管理人员则能根据本文档评估软件系统开发大致的时间流程并制定详尽的开发计划。

1.2 范围

（1）软件系统的名称：顺风打车系统

（2）文档内容：本体系结构设计文档分析了顺风打车系统的功能、

系统设计的总体结构以及相应的设计策略。本软件体系结构设计文档能够满足系统的功能和非功能性需求(主要为质量、可信赖性要求),并且分析了在未来的升级和维护中可能遇到的问题以及相应的解决方案。

1.3 定义，首字母缩略字及缩写

（1）顺风打车系统：一个可以方便打车用户打车叫单，方便司机进行接单，为司机提供路线导航，实现订单交易的打车软件系统。

（2）功能需求：功能需求是对未来软件的服务功能进行的描述。功能需求更关注系统的输入、加工（业务流程）直到输出的情况。

（3）非功能需求：非功能需求是指功能以外的需求描述，主要是系统的一些关键的特性，例如可靠性、响应时间、存储空间、I/O 设备的吞吐量、接口的数据格式等。

（4）用户角色：用户角色是指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、便于统一描述的众多用户个体的集合。

（5）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视图。用例图（User Case）是被称为参与者的外部用户所能观察到的系统功能的模型图，呈现了一些参与者和一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。

(6)概念级体系结构:描述系统的主要设计元素和元素之间的关系。

（7）模块级体系结构：按两两正交模块的结构组织系统，即，功能

分解和层次分解。

（8）运行时体系结构：描述系统的动态结构。

（9）代码级体系结构：描述开发环境中的源代码、二进制代码、开发库的组织形式。

（10）ATAM：（Architecture Tradeoff Analysis Method）体系结构折中分析方法。这是SEI开发的通过手工评估和确认体系结构的方法。ATAM将场景分为三种类型：

- 用例场景(use case scenarios)

用例场景描述系统的典型使用，可以启发出实际的应用情况；

- 增长性场景(growth scenarios)

增长性场景涵盖了对系统的预期更改的场景；

- 探索性场景(exploratory scenarios)。

探索性场景可以涵盖极端的修改情况,期望对系统进行“压力”确认。

（10）C/S模式：即Client/Server（客户机/服务器）结构，是大家熟知的软件系统体系结构，通过将任务合理分配到Client端和Server端，降低了系统的通讯开销，需要安装客户端才可进行管理操作。

客户端和服务器端的程序不同，用户的程序主要在客户端，服务器端主要提供数据管理、数据共享、数据及系统维护和并发控制等，客户端程序主要完成用户的具体的业务。

开发比较容易，操作简便，但应用程序的升级和客户端程序的维护较为困难。

（11）B/S 模式：即 Browser/Server（浏览器/服务器）结构，是随着 Internet 技术的兴起，对 C/S 结构的一种变化或者改进的结构。在这种结构下，用户界面完全通过 WWW 浏览器实现。

客户端基本上没有专门的应用程序，应用程序基本上都在服务器端。由于客户端没有程序，应用程序的升级和维护都可以在服务器端完成，升级维护方便。由于客户端使用浏览器，使得用户界面“丰富多彩”，但数据的打印输出等功能受到了限制。为了克服这个缺点，一般把利用浏览器方式实现困难的功能，单独开发成可以发布的控件，在客户端利用程序调用来完成。

1.4 参考

[1] 王安生，《软件工程化》[M].北京:清华大学出版社，2014

[2] 陈宇昂. 软件体系结构深入探索[J]. 2009, 25(10):73-73.

[3] 李书杰, 李志刚. B/S 三层体系结构模式[J]. 河北联合大学学报(自然科学版), 2002, 24(s1):25-28.

[4] 徐建, 荆文娟, 严悍,等. 一种软件体系结构风险评估方法[J]. 南京理工大学学报(自然科学版), 2010, 34(5):680-685.

1.5 概述

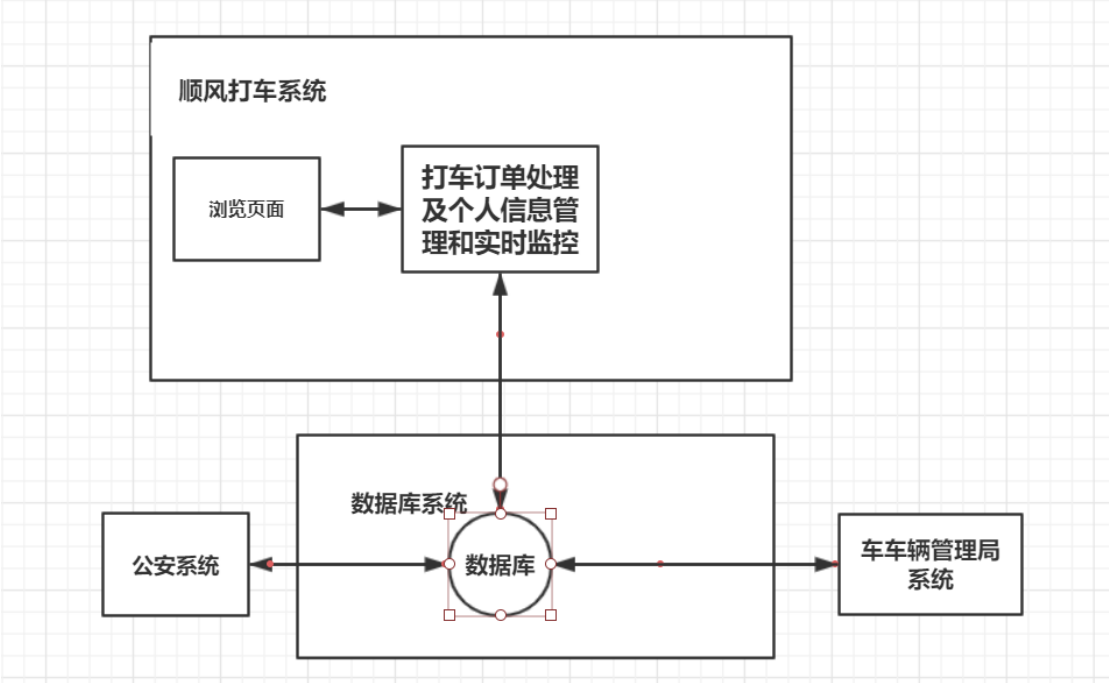
本文档将依次介绍：项目相关信息、体系结构需求、解决方案、系统质量分析和评价等。

2.项目相关信息

2.1 产品描述

此顺风打车系统系，实质上是个基于 web 和 App 服务的信息管理系统，目的是为了方便出行的用户尽快叫到车辆，提高司机们的接单效率，方便司机进行接单，通过路线规划来提高司机的送车效率，同时通过增加了乘客们出行的安全性，减少黑车司机的作案可能。系统的用户分为四类：乘客（打车用户）、司机、系统管理员、业务监测人员、特殊情况处理员。乘客登系统后可以进行出行叫单，可以选择订单的种类（拼车、快车、出租、优享等等）、可以实时查看订单的情况，当订单被司机接收后可以查看司机车辆的位置以及行车路线，可以查看系统对自己行程的路线规划、到达目的地后用户可以通过系统进行订单支付，并对司机进行评价。司机登录系统后可以选择接单，可以查看系统给出的路线导航信息，可以通过手机号联系乘客，送达乘客之后可以通过系统收取费用。业务监测员登录系统之后，可以分析系统会将信息呈现在屏幕上，业务检测员可以进行详细的流量信息查看，如果发现某段时间的情况与原来这个时间段订单情况差别很多，需要进行详细分析。当有用户进行呼救时，特殊情况处理员在接到求助时，可以查看该用户所在的实时位置以及订单的详细进行意外情况处理，如果情况严重可可以呼叫公安部门进行帮助。系统管理员对业务监测员、特殊情况处理员、乘客、司机的个人信息和使用权限进行管理。

系统管理员能管理司机、乘客、业务监测员、特殊情况处理预案的权限和账户信息。



2-1 系统块图

2.2 产品角色和功能

（1）司机

- 对个人的账户信息进行管理。
- 按照学期查询自己的体育理论成绩、项目测试成绩、平时成绩。

（2）乘客（打车用户）

- 对个人的账户信息进行管理。
- 录入所教科目的所有学生的理论成绩和项目测试成绩，之后向教务员提交审核。
- 按规定的格式录入平时成绩的计算规则。
- 可以查看所有自己所教课程的学生的成绩，并且可以按照班级查

看所教课程的各项统计数据，包括平均分、最高分、最低分、及格率等。

- 若发现成绩有误，可以向教务长申请权限修改已录入的学生成绩。

（3）业务检测员

- 对个人的账户信息进行管理。
- 业务检测员使用 web 登录系统进行监控，在使用之前必须申请权限授予等待系统管理员的权限授予过后才可以从 web 登录系统。
- 业务监测员登录系统之后，可以实时观察当前的订单流量，智能分析系统会将信息呈现在屏幕上，业务检测员可以进行详细的流量信息查看，如果发现某段时间的情况与原来这个时间段订单情况差别很多，需要进行详细分析
- 业务检测员需要对实时路况进行监管，如果发现某段路况非常拥挤而导航都往这个路段进行导航，需要记录下来，想系统维护人员说明。

（4）系统管理员

- 对个人的账户信息进行管理。
- 管理账户信息。系统管理员可以对所有类型的用户（学生、教师、教务长）进行权限的授予撤回以及账户的增删和信息的修改。

（5）特殊情况处理员

- 对个人的账户信息进行管理。
- 特殊情况处理员可以接收司机和打车用户的呼叫处理，处理之后需要将处理的情况记录在案。

- 当有用户进行呼救时，特殊情况处理员在接到求助时，可以查看该用户所在的实时位置以及订单的详细进行意外情况处理，如果情况严重可可以呼叫公安部门进行帮助

2-2 系统的结构功能图

3.体系结构需求

3.1 体系结构定义及关键指标

1.体系结构的定义： 软件体系结构是具有一定形式的结构化元素，即构件的集合，包括处理构件、数据构件和连接构件。处理构件负责对数据进行加工，数据构件是被加工的信息，连接构件把体系结构的不同部分组合连接起来。这一定义注重区分处理构件、数据构件和连接构件，这一方法在其他的定义和方法中基本上得到保持。

2.关键指标说明

2.1 从软件需求角度

功能性：体系结构体现系统所期望需求功能的能力。

性能：是指按该体系结构实施后的系统的响应能力。影响性能的体系结构因素包括计算在组件上怎样分配。组件(内部进程和交互进程)间的通信模式等。

可靠性：可靠性是按该体系结构实施下的软件系统在系统发生错误。或在意外或错误使用的情况下．采用体系结构策略维持软件系统功能特性的基本能力。

可用性：是指按该体系结构实施的系统能够正常运行的时间比例。影响该指标的因素包括冗余构件和故障检测构件。

安全性：是指按该体系结构实施的系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。

扩充性：系统的体系结构应满足系统生存期间额外的要求．允许

增加新的构件或结构以满足系统新的需求的能力。

死锁性：在体系结构的设计中。是否能够避免构件间的交互被”卡住”,不能继续向前的状况。

适应性：体系结构是否有适应用户变化而变化的能力。

2.2 从软件实现角度

可扩展性：是指该体系结构支持用新特性来扩展软件系统．允许开发人员在不影响客户的情况下替换构件以及支持把新构件集成到现有的体系结构中的能力。

结构重组：是指该体系结构根据开发需要重新组织软件系统的构件及构件问的关系的能力。

移植性：是指该体系结构实施下.软件系统适用于多种硬件平台、用户界面、操作系统、编程语言或编译器的能力。

可变性：是指体系结构经扩充或变更而成为新体系结构的能力。

可集成性：是指按此体系结构构成的系统能与其他部件协作的容易程度。

互操作性：是指该体系结构实施下。系统与其他系统或自身环境相互作用的能力。

可测试性：是指软件体系结构可以为错误探测和改正。以及调试代码和组件的Ｉ临时集成给予支持的能力。

可理解性：是指软件体系结构的概念和描述能够全面表达和被深刻理解的程度．反映了该体系结构支持软件设计者之间、设计者与用户之间可以快速方便地交流知识、经验和新设计思想的能力。

可重用性：是指体系结构的框架或映射到体系结构中的构件可以重用体系结构库的结构或构件库的构件的程度。

效率性：是指按该体系结构实施的系统在运行时存储空间、运行时间、吞吐量等对系统的影响程度。

可维护性：按该体系结构实施的系统在错误发生后”修复”软件系统的能力。

2.3 从软件工程管理的角度

部件无关：是指体系结构中部件无需了解其它部件的存 在就能独立工作的特性。

风险性：按此体系结构实施的技术和技能的风险度的高低。

复用性：反映该体系结构的抽象性和通用性的程度。

可度量性：按该体系结构实施，系统的开发进度，人力、资金、资源调配可以估计的能力。

正交性：是指反映体系结构中同一层次的构件之间不存在相互调用的状况。

3.2 体系结构用例

本说明书的用例是从使用者的角度出发，描述用户所期望的与整个运行系统的交互 [1]。根据《软件需求分析说明书》对本系统用户需求的分析以及对用例角色的陈述，本体系结构说明书将用户的需求与系统进行交互，设计体系结构用例如表 1

表 1.体系结构表

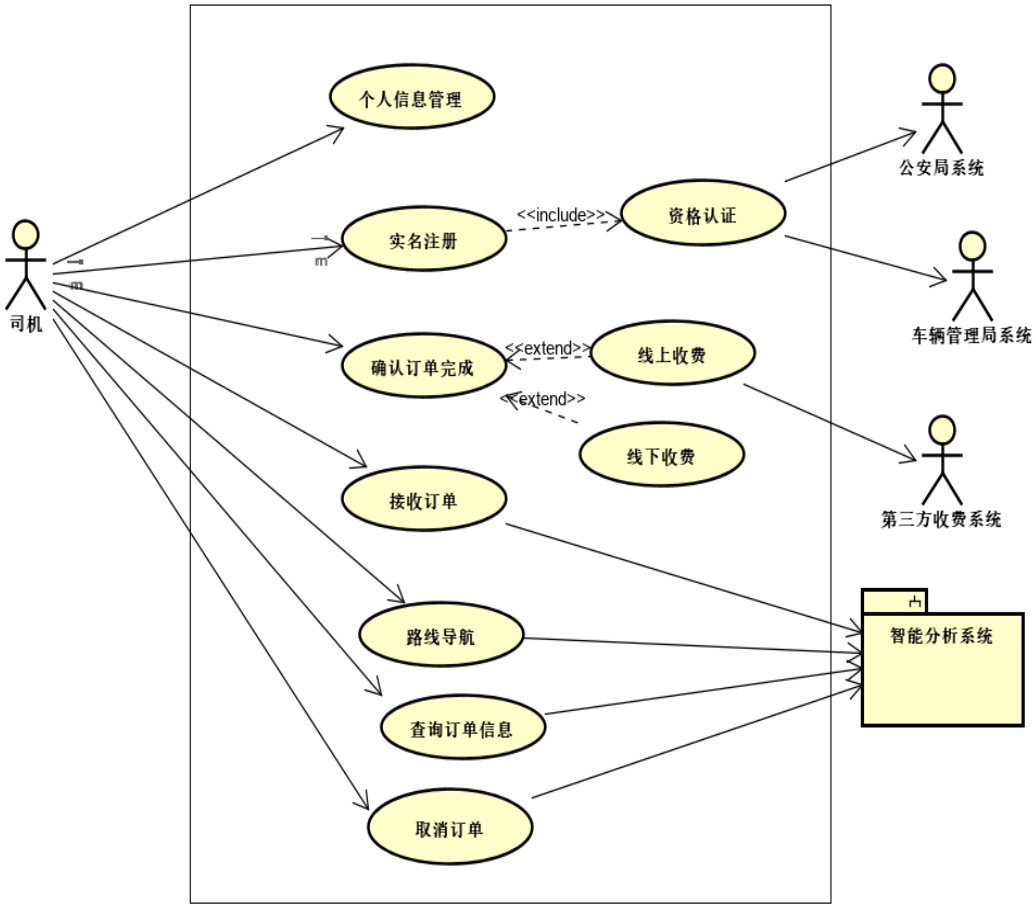
角色	*用途*
软件开发人员	1. 清楚知道软件的整体结构。 2. 清楚知道各模块间的接口，方便编写。 3. 清楚知道自己所要完成的功能。
产品经理	1. 知道软件系统的规模。 2. 根据体系结构安排工作给各开发小组。 3. 决定开发策略。 4. 规划工期进度安排
测试人员	1. 明确系统的性能指标。 2. 明确系统的组织结构。 3. 编写测试用例的依据。
甲方人员	1.对该体系结构文档给出评价。 2.验证其是否能满足自己的需求。 *3.项目验收的标准。*
司机和乘客	*1.*对该体系结构文档提出修改建议。

3.2.1. 角色

- 乘客
- 司机
- 业务监测员
- 特殊情况处理员
- 管理员

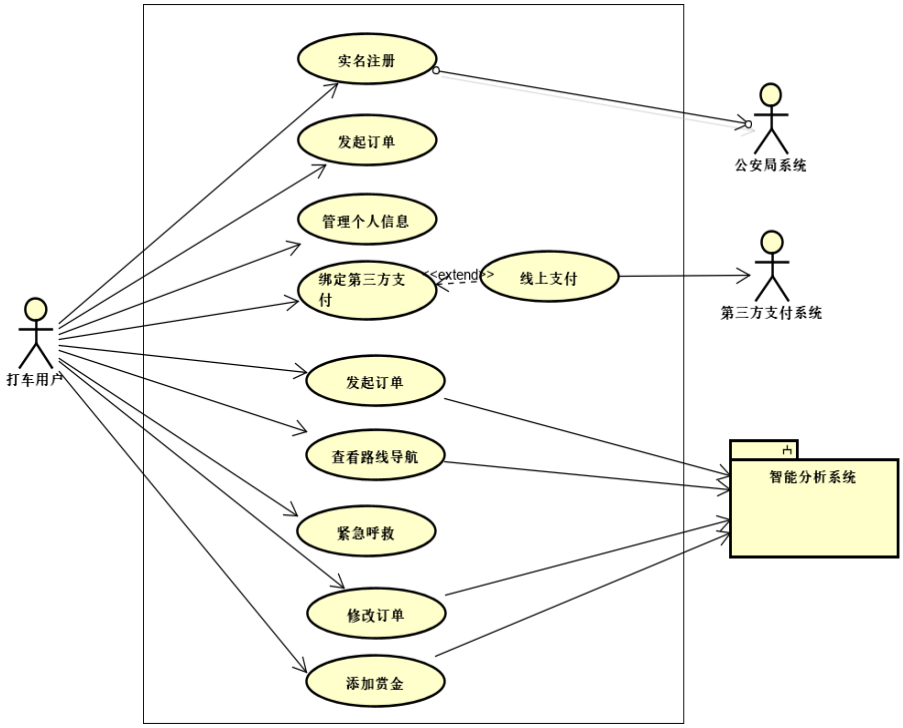
3.2.2 用例图

司机交互模块



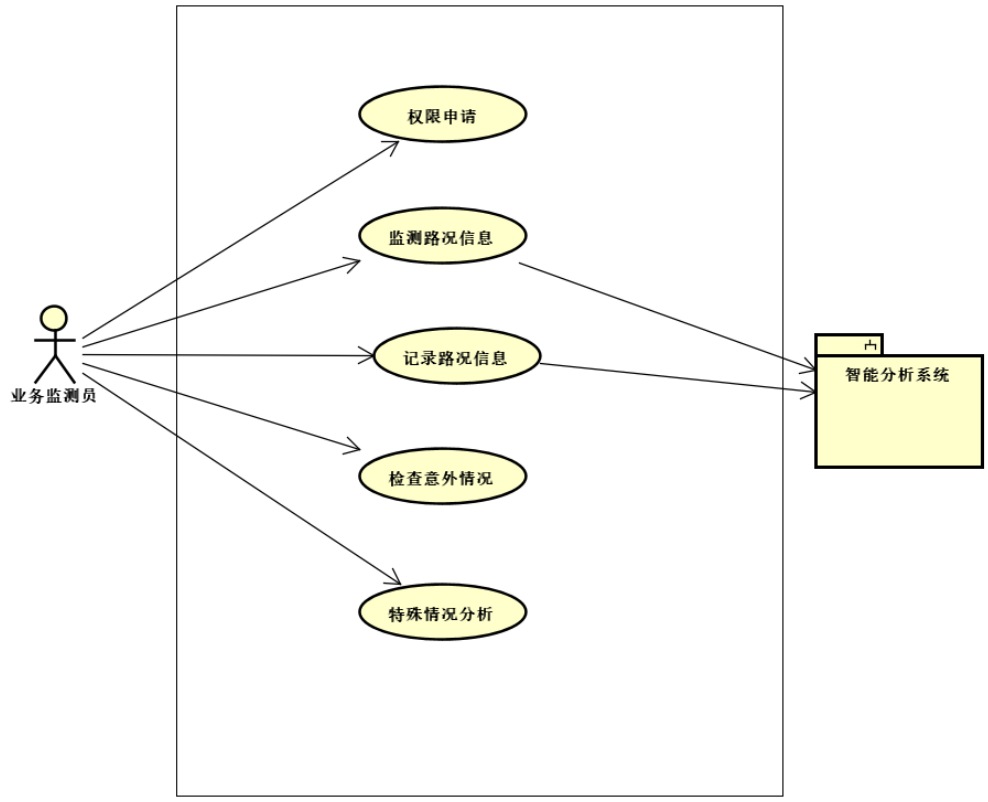
3-1 司机交互模块用例图

打车用户交互模块



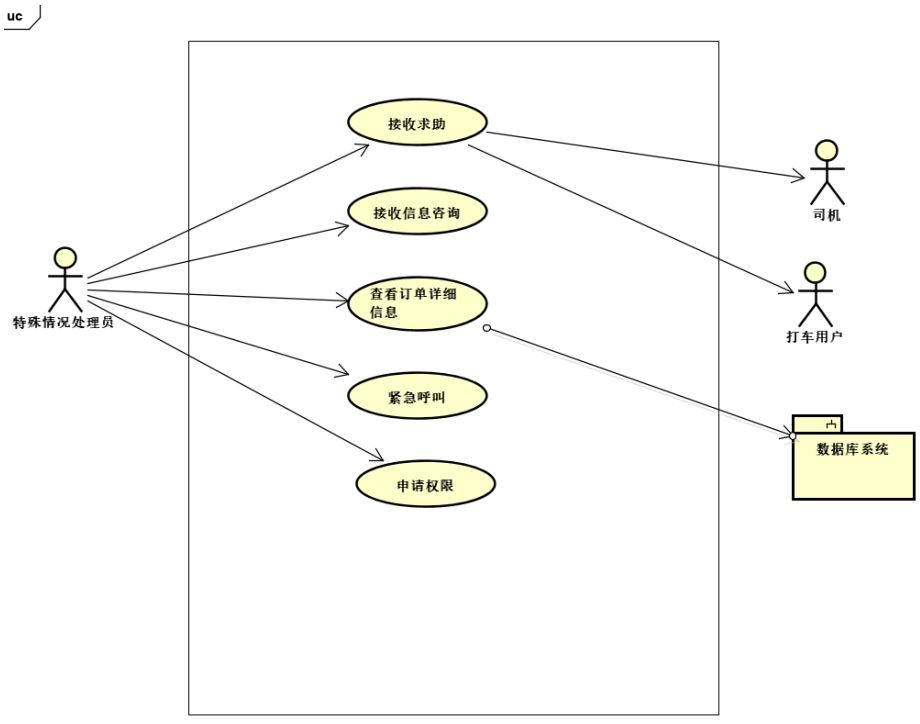
3-2 打车用户交互模块用例图

业务监测员交互模块



3-3 业务监测员交互模块用例图

特殊情况处理员交互模块



3-4 特殊情况处理员交互模块用例图

在需求报告中已经给出了需求分析时考虑的用例，设计体系结构时考虑一些用例需要进行修改，修改的用例如下：

实时呼叫

名称：实时呼叫
范围：车辆呼叫子系统
级别：用户目标级别
参与者：乘客
涉众及其关注点：
乘客：能看到不同类型车辆的价格估算，能成功发起呼叫
司机：能依照自己的车辆类型收到乘客发起的呼叫
前置条件：乘客登录了账号，且信息完善
成功保证：乘客的呼叫订单被加入到系统队列中，且被司机抢单抢到
主要成功场景：
乘客发起呼叫，司机抢到乘客发起的呼叫，司机联系乘客确认上车地点正确，乘客等待司机，司机准时到达约定地点，乘客上车。
扩展：
1. 提前取消订单的情况：乘客发起呼叫，司机抢到乘客发起的呼叫，乘客 2 分钟内取消订单，司机无偿放弃订单。
2. 超时取消订单的情况：乘客发起呼叫，司机抢到乘客发起的呼叫，乘客 2 分钟后取消订单，司机获得赔偿并放弃订单
3. 司机迟到的情况：乘客发起呼叫，司机抢到乘客发起的呼叫，司机联系乘客确认上车地点正确，乘客等待司机，司机迟到，乘客无偿取消订单
4. 乘客迟到的情况：乘客发起呼叫，司机抢到乘客发起的呼叫，司机联系乘客确认上车地点正确，司机准时到达约定地点，乘客迟到，司机无偿取消订单

预约呼叫

名称：预约呼叫
范围：车辆呼叫子系统
级别：用户目标级别
参与者：乘客
涉众及其关注点：
乘客：除了实时呼叫中的关注点外，还需关注预约乘车时间
司机：根据自己的需要接收预约订单
前置条件：乘客登录了账号，且信息完善，预约信用良好
成功保证：乘客的呼叫订单被加入到系统队列中，且有司机愿意接受预约
主要成功场景：
乘客发起预约，司机接受乘客的预约，二者按提前约定的时间地点准时见面，乘客上车。
扩展：

1. 没有司机接受的情况：乘客发起预约，在预约时间到达前一小时内仍无司机接受预约，则加大预约的推广范围，让更多的司机看到该乘客的预约

抢单

名称：抢单
范围：车辆呼叫子系统
级别：用户目标级别
参与者：司机
涉众及其关注点：
司机：能收到乘客发起的订单
乘客：能被司机抢到自己的订单
前置条件：司机在乘客的接单范围内
成功保证：在多名司机同时抢单的竞争中胜出
主要成功场景：
司机收到乘客发起的订单，司机点击抢单，在竞争中获胜，抢单成功，准备去接该乘客

结束订单

名称：结束订单
范围：车辆呼叫子系统
级别：用户目标级别
参与者：司机
涉众及其关注点：
司机：到达乘客行程的终点时结束订单要求乘客付款
乘客：到达终点时要求司机及时结束订单
前置条件：在合理的结束订单的范围内，或乘客同意结束订单
成功保证：在多名司机同时抢单的竞争中胜出
主要成功场景：到达终点范围，结束订单
扩展：
1. 未到终点范围，但乘客同意结束订单
2. 未到终点范围，司机擅自结束订单，乘客举报，客服核实，乘客获得赔偿，管理员对司机做出处罚

支付

名称：支付
范围：支付子系统
级别：用户目标级别
参与者：乘客
涉众及其关注点：
乘客：能成功支付车费
司机：乘客能成功支付车费
前置条件：订单已结束，乘客的移动支付账户处于激活状态，且余额充足

成功保证：移动支付平台返回成功信息
主要成功场景：司机结束订单后，乘客端跳转到支付界面，乘客选择支付方式，跳转到移动支付平台的支付界面完成支付，然后返回小精灵打车。支付款先到达平台，然后由平台打给司机。
扩展：
1. 乘客允许自动支付的情况：司机结束订单后，乘客端自动完成支付，支付款到达平台。
2. 乘客支付失败（如余额不足）：司机结束订单后，乘客端准备支付，但遇到状况支付失败，支付款未到达平台，则将保留该订单，并定时提醒乘客完成支付。

评价司机

名称：评价司机
范围：评价子系统
级别：用户目标级别
参与者：乘客
涉众及其关注点：
乘客：能顺利地完成评价过程
管理员：能对自己所管理的司机的评价
前置条件：乘客所评价的司机必须为该乘客服务过且完成了订单
成功保证：评价成功且被管理员收到
主要成功场景：乘客对司机做出星级评价、文本评价和标签评价，司机无法看到内容评价，管理员可以看到完整的评价。

3.3 各相关方对体系结构的要求

1）**司机**希望自己可以随时随地查询自己的到达乘客的最近路线，并且这个过程应该是方便快速的。因此这就限制了体系结构应该是采用 Browser/Server 的结构，而不能是 Client/Server 的结构。

2）程序为了适应未来停车位信息以及费用计算模式的变化要求系统具有高可修改性、可移植性和可重用性，这要求体系结构要采用分层设计的思想。

3）每一个司机只能看到自己的停车费的信息，并且，每一个管理员所管理的车位不同，看到的界面也是不一样的。因此需要体系结构可

以满足很容易的修改一些用户界面的要求，比如 MVC 模式可以很好的满足这一要求。

4）目前，随着大规模的网络技术迅速的发展使用，在人们日常的工作、生活过程中，产生了海量的数据，为了能够让人们有效的利用这些数据，本地数据库已经无法满足人们的多种多样的需求，因此，在工作、学习和生活过程中，必须开发基于网络的分布式应用系统，提高海量数据信息的使用效益。分布式体系架构是实现分布式应用系统的基本，为了能够使得分布式应用系统实现高性能访问和处理，诸多科学工作者对其进行了研究。

5）系统开发、测试、维护人员要求

体系结构的设计应与部件无关、低风险性、高可复用性高、可度量性、高可移植性、高可维护性。

与部件无关：是指体系结构的部件可相互独立地工作，无需了解其他部件的具体实现原理

低风险性：按此体系结构实施的技术和技能的风险度较低。

可复用性：体系结构设计抽象，且具有高通用性

可度量性高：按该体系结构实施，系统的开发进度，人力、资 金、资源调配可以估计的能力。使项目管理人员能更好地进行项目进度的管理和安排

高移植性：统软件的接口易改造，可以比较容易地转移到其他计算机上使用；同时系统使用了跨平台的 Java 语言来编写并且使用了开源库和复用了一些开源项目的代码，因此可以比较方便地移植到不同平

台上。

高可维护性：系统的构建结构合理，采用“高内聚、低耦合”的原则，可以比较方便地进行修改、升级、测试、维护。

6）**用户要求**

系统应具有高可用性

- 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的扁平化设计。

3.4 约束条件

1）系统要求在 12 个月内完工。
2）可以满足随时随地查询使用的原则，因此要求采用 B/S 架构。
3）硬件、软件、运行环境和开发环境方面的条件和限制：CPU:双核 T1600 以上；

内存：1G 以上；硬盘：2G 以上；编译程序：Java，Mysql；操作系统：支持

Win7、Win8、Win10 等主流 Windows 操作系统以及 Android9,8 和 IOS12,13 系统。
4）用 JAVA 语言实现 “顺风打车系统”。

5）系统必须能支持 35000 个并发用户对系统发出服务请求，且其访问速度不得大于 10s；
6）法律和政策方面的限制：开发此软件时，将严格按照有关法律和政策执行。

3.5 非功能需求

3.5.1 性能表现

1.时间性能：

(1)系统执行速度

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。
- 系统应当能够计算打车用户订单的出发地附近的最佳司机信息（可以接单的司机信息），一个流程的响应速度最佳为 b4s,平均为 a4s，不超过 m4s。系统应 当可以同时接受 10000 人同时操作。
- 系统应当能够赋予特殊情况处理员查看订单信息的权限，一个流程的 响应速度最佳为 b5s,平均为 a5 s， 不超过 m5s。系统应当可以同时接受 10000 人同时操作。

- 系统应当能够计算体育成绩（包括计算平时成绩和总成绩），其响应速度最佳为 b4s,平均为 a4s，不超过 m4s。
- 系统应当能够赋予特殊情况处理员查看订单信息的权限，一个流程的 响应速度最佳为 b5s,平均为 a5 s， 不超过 m5s。系统应当可以同时接受 10000 人同时操作。
- 系统应当能够更新车位信息和订单信息，一个流程的响应速度最佳为 b3s,平均为 a3 s，不超过 m3s。系统应当可以同时接受 10000 人同时操 作。

2.空间性能：

（1）数据库容量能够存储至少 50000 条学司机实体童虎，但考虑到可能订单的增加和乘客用户的扩张,本系统的数据库容量设置为能够存储 150000 条乘客实体用户。

（2）操作占用内存的限制

- 系统应当能够查看和修改四种用户信息，每次操作占用内存不超过 s1。
- 系统应当能够查询订单详细信息信息，每次操作占用内存不超过 s2
- 系统应当能够更新路况信息和车辆所在位置 i 西南西，每次操作占用内存不超过 s3。
- 系统应当能够根绝位置信息和订单信息派单，每次操作占用内存不超 过 s4。

（3）系统有足够大的缓存空间，保障系统运行流畅。

3.5.2 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，详细分为如下：

1.系统的可连续运行时间在一年以上，并且每次发生故障或者宕机后，能够在半小时内恢复正常。

2.故障恢复能力。每次发生故障或者宕机后，能够在半小时内恢复正常。

3.容错能力。系统具有一定的容错和抗干扰能力，在非硬件故障或非通讯故障时，系统能够保证正常运行。如用户在系统中输入不规范时，不会引起系统的故障，且系统会给出提示信息，并限定不正确输入的次数。

4.数据负载能力。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。

5.云服务的可靠性：

云服务的可靠性主要分为以下几个方面。

一是物理服务器的可靠性。服务器需要 7×24 小时不间断地运行，因此需要完善的防灾措施，使其远离火灾、地震等灾害。

二是云服务逻辑上的可靠性。云服务的逻辑中不能存在死循环、死锁、时间过长的响应等影响服务可靠性的因素。

三是网络的可靠性。与客户端相同，云服务端也需要一些工作来保证网络请求和响应的正确性。例如客户端在收到服务器发送的响应前断网了，应该在服务器缓存该响应。

3.5.3 可用性

- 该系统的可使用性体现在它可以支持多操作系统(至少包含 Andorid,IOS,Windows,Unix,Linux)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的扁平化设计。

3.5.4 密安性

系统应当能够保证用户信息不泄露，系统配置文件和数据库存储文件应该进行加密处理

(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。

(6) 财产密安性：对于移动支付而言，财产的密安性本质上就是数据的密安性。但财产的密安性应当比普通密安性更重要，威胁财产安全一般而言比威胁普通数据安全的损失要大。采用去中心化的区块链技术进行交易记录的存储是保证财产密安性的最基本的方式。这其中将涉及一些加密技术。

(7) 数据库的数据密安性：从数据库的角度来说，软件使用的过程中存在大量的数据库的增删改查操作，而每个用户的增删改查权限应该

是不同的，应该对用户进行分组，每个用户组只能对特定的数据表特定的字段进行特定的操作。。

3.5.5 可维护性

1.软件的可维护性是指改进软件的难易程度。系统部件遵守“高内聚，低耦合”原则。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。

2.系统应该能够容易诊断出存在的缺陷和失效原因，容易识别出待修改部分的可能性或能力。开发人员应当记录开发过程日志，以便备份追踪。

3.系统应当能够保证开发过程的代码、设计和文档容易修改，代码应当结构清晰且有较详细的注释，设计文档详细明确

4.从数据库的角度来说，本软件数据的可维护性包括以下几个方面。

1)是要求数据的物理存储是可以更换的。如果一台云数据库出现问题，可以及时地将该云数据库中地数据拷贝到另一台云数据库中

2)是要求 DBMS 是可以更换的，即数据地逻辑存储是可以更换的。

3)是要求数据的视图层是可以更换的。

3.5.6 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,客户端应该支持Android,IOS,鸿蒙系统等平台。

2.共存性：系统应当能够和其他软件共存于一个平台上，不存在冲突的软件。

3.可替换性：系统可被容易地卸载，也易被高版本的系统替换。

4.接口易改造：可方便地移植到不同的设备上。

5.云服务应采用 MVC 的架构，模型层与数据库层直接联系，因此可用统一的方式编写,控制层在设计阶段就已确定,也可统一编写。如此，视图层可以使用不同的框架进行编写，保证其可移植性

3.6 设计需遵循的标准

该软件系统必须遵循以下设计原则：

3.6.1 实用性原则

顺风打车系统的设计过程中,要充分考虑各个高校的实际情况进行业务逻辑的设计，以保证最终开发出来的软件系统切实可用，可以良好地运行。

3.6.2 稳定性原则

顺风打车系统能 24 小时提供不间断的服务，要保证用户在任何时刻都能使用到我们的平台来他们所需的功能。因此，在设计时要考虑到任何可能导致服务中断的原因，为用户提供稳定的服务。

系统设计时应有系统故障的应急预案，当主系统发生故障时，应保证可从主系统平滑地切换到备份系统,尽可能使用户感知不到系统发生的故障，把故障的损失降到最低。

3.6.3 复用性原则

组件复用是顺风打车系统设计中必须遵守的原则。对可重用的组件进行统一包装，包括系统级的应用组件和应用级的服务组件。其中，系统级的应用组件主要考虑应用服务器和数据库的广泛适用性，对各种企业级服务进行重新包装，以便于提高架构的平台独立性。以使后续能够为不同的高校开发出适合他们实际情况的顺风打车系统系统。

3.6.4 扩展性原则

顺风打车系统必须具有良好地可扩展性。随着移动互联网的兴起，越来越多的用户选择在通过小程序,快应用等新型模式使用互联网。因此，顺风打车管理系统应能在未来扩展到支持小程序,快应用等平台，以满足用户的需求。

3.6.5 安全性原则

顺风打车系统架构的设计将在成熟稳定的硬件环境和应用软件基础上，通过网络、系统、应用、备份恢复、安全控制机制、运行管理监控等手段来保障系统的安全运行。

1.应充分、全面、完整地对系统的安全漏洞和安全威胁进行分析；

2.在网络发生被攻击、破坏事件的情况下，应尽可能地快速恢复网络信息中心的服务，减少损失。应建立安全防护机制、安全检测机制

和安全恢复机制。

3.建立合理的实用安全性与用户需求评价与平衡体系。

4.系统是一个庞大的系统工程，其安全体系的设计必须遵循一系列的标准，这样才能确保各个分系统的一致性，使整个系统安全地互联互通、信息共享。

5.技术与管理应相结合原则。应将各种安全技术与运行管理机制、人员思想教育与技术培训安全规章制度建设相结合。

6.统筹规划，分步实施原则。

7.等级性原则。将系统进行分为不同的等级，包括对信息保密程度分级，对用户操作权限分级，对网络安全程度分级，对系统实现结构的分级，从而针对不同级别的安全对象，提供全面安全算法和安全体制。

8.动态发展原则

9.易操作性原则

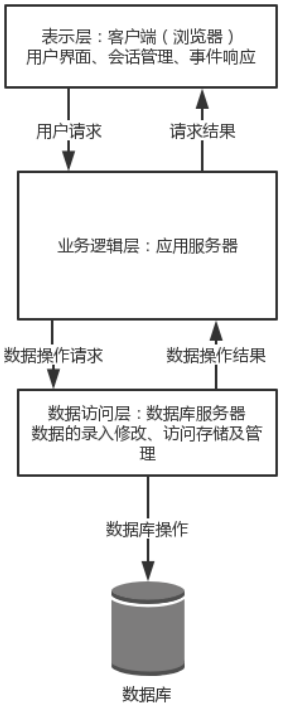
安全措施需要人为去完成，如果措施过于复杂，对人的要求过高，本身就降低了安全性。其次，措施的采用不能影响系统的正常运行。

4.解决方案

4.1 相关的体系结构模式

4.1.1 三层 B/S 体系结构概述

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，是从传统的 C/S 发展起来的计算方式。C/S 是松耦合的系统，通过消息传递机制进行对话，由客户端发出请求给服务器，服务器进行相应的处理后经传递机制送回给客户端；B/S 模式则把 C/S 模式的服务器进一步深化，分解成应用服务器和多个数据库服务器，仅保留其表示功能，从而称为一种由表示层（Browser)、功能层（Web Server)、数据库服务层(DataBase Server)构成的三层分布式结构。如下图所示：



4-1 三层 B/S 体系结构体示意图

图中,客户端只有Browser,一般没有应用程序,借助于Java applet、VBScript、JavaScript、ActiveX 技术可处理一些简单的客户端处理逻辑，显示用户界面和 WebServer 端的运行结果。它向 URL 所制定的 Web 服务器提出服务申请，Web 服务器对用户进行身份认证之后，用 TCP/IP 协议把所需的文件资料传送给用户，客户端只是接收文件资料，并显示在浏览器上。

服务器端由 W eb server 和 DATABASE server 构成，W eb server 负责接受远程或本 地的数据查询请求，然后运行服务器脚本，借助于 CG I 、 ADO、 API 、 JDBC 等中间部件把数 据请求通过发送到 DB server 上以获取相关数据，再把结果数据转化成 HTML 及各种脚本传回客户的 Browse 。

DB server 负责管理数据库，处理数据更新及完成查询要求、运行存储过程。可以是 集中式的也可以是分布式的。

由图中可以看出， Browse 与 Web Server 之间的关系就类似于主机／终端结构中两 者之间的关系，而 W eb Server 与 DB Server 之间的关系就像 Client／Server 结构中两者之间 的关系。 在三层结构中，数据计算与数据处理集中在中间层，即 Web Server 这一层，由于中间层的服务器的性能容易提升，所以在 Internet 下的三层结构可以满足用户的需求。但这种结构对数据库服务器提出了高要求。从管理的角度来看，程序代码的维护、数据库的备 份虽然可以在服务器端执行，但这种三层结构对网络带宽是有一定要求的，因为客户端每 次要求获取的数据和反馈信息都要通过网络与服务器联系。

4.1.2 三层B/S体系结构的优点

1.开放的标准

Client / Server 所采用的标准只要在内部统一就可，它的应用往往是专用的。Browser / Server 所采用的TCP / IP、 HTTP 等标准都是开放的、非专用的，是经过标准化组织所确定的而非单一厂商所制定，保证了其应用的通用性和跨平台性。同时，标准化使得B / S模式 可直接接入Internet ，具有良好的扩展性、伸缩性，可从不同厂家选择设备和服务。

2.分布计算的基础结构

多层的B / S 应用可以更充分的利用系统资源，在大型的联机应用中，数据库面临的客 户数量是非常庞大的，使用传统的客户 / 服务器模式可能根本无法胜任。例如，可能有上 千个客户机在同时运行，需要访问数据库。如果它们的请求都直接传递到数据库服务器 上，就必须要有非常强大的硬件支持。通过中间层的缓冲，连接数据库的用户数大大减少。虽然增加了应用服务层，并不会使系统的性能和可靠性降低。因为在动态分布式计算系统中，客户端程序不必要确切指出应用服务的网络地址，如果应用服务器超负荷，通过统一的管理程序调度将请求转移到其他应用服务器上来消除瓶颈。

3.较低的开发和维护成本

Client / Server 的应用必须开发出专用的客户端软件，无论是安装、配置还是升级都需要在所有的客户机上实施，极大地浪费了人力和物力。Browser / Server 的应用只需在客户端装有通用的浏览器

即可，维护和升级工作都在服务器端进行，不需对客户端进行任何改变，故而大大降低了开发和维护的成本。

4.使用简单，界面友好

Client／ Server 用户的界面是由客户端软件所决定的，其使用的方法和界面各不相同， 每推广一个Client／ Server 系统都要求用户从头学起，难以使用。Browser／ Server 用户的界面都统一在浏览器上,浏览器易于使用、界面友好,不须再学习使用其它的软件，一劳永逸 的解决了用户的使用问题。

5.系统灵活

Client／ Server 系统的三部分模块中有一部分需改变就要关联到其它模块的变动，使 系统极难升级。Browser／ Server 系统的三部分模块各自相对独立，其中一部分模块改变时， 其它模块不受影响，应用的增加、删减、更新不影响用户个数和执行环境，系统改进变 得非常容易，且可以用不同厂家的产品来组成性能更佳的系统。

6.保障系统的安全性

在Client／ Server 系统中由于客户机直接与数据库服务器进行连接，用户可以很轻易的改变服务器上的数据，无法保证系统的安全性。Browser／ Server 系统在客户机与数据库 服务器之间增加了一层 W eb 服务器，使两者不再直接相连，通过对中间层的用户编程可实现更加健全、灵活的安全机制。客户机无法直接对数据库操纵，有效地防止用户的非法 入侵。

7.信息共享度高

Client / Server 系统使用专用的客户端软件，其数据格式为专用格式文件。Browser / Server 系统使用 HTML， HTML 是数据格式的一个开放标准，目前大多数流行的软件均 支持 HTML，同时 MIME 技术使得 Browser 可访问多种格式的文件。

8.广域网支持

Client / Server 系统是基于局域网的，而 B rowser / Server 系统无论是 PSTN、 DDN、帧中 继， X.25 、 ISDN，还是新出现的 CATV、 ADSL， B / S 结构均能透明的使用。

4.1.3 三层 B/S 体系结构对质量属性的影响

质量属性	对质量的影响
性能	这种结构的性能已经得到了证明。关键的问题是要考虑每个服务器支持的并发线程数量、各层之间的连接速度以及数据传递的速度。对于分布式系统来说，降低了为完成每个需求所需的层与层之间的调用时间。
可靠性	通过中间层的缓冲，连接数据库的用户数大大减少。虽然增加了应用服务层，并不会使系统的性能和可靠性降低。因为在动态分布式计算系统中，客户端程序不必要确切指出应用服务的网络地址，如果应用服务器超负荷，通过统一的管理程序调度将请求转移到其他应用服务器上来消除瓶颈。
可用性	Browser / Server 用户的界面都统一在浏览器上，浏览器易于使用、界面友好，不须再学习使用其它的软件，一劳永逸 的解决了用户的使用问题。
密安性	Browser / Server 系统在客户机与数据库 服务器之间增加了一层 W eb 服务器，使两者不再直接相连，通过对中间层的用户编程可实现更加健全、灵活的安全机制。客户机无法直接对数据库操纵，有效地防止用户的非法 入侵。
可维护性	Browser / Server 系统的三部分模块各自相对独立，其中一部分模块改变时， 其它模块不受影响，应用的增加、删减、更新不影响用户个数和执行环境，系统改进变 得非常容易，且可以用不同厂家的产品来组成性能更佳的系统。

可移植性	Browser / Server 系统使用 HTML， HTML 是数据格式的一个开放标准，目前大多数流行的软件均 支持 HTML，同时 MIME 技术使得 Browser 可访问多种格式的文件。
可伸缩性	Browser / Server 所采用的 TCP / IP、 HTTP 等标准都是开放的、非专用的，是经过标准化组织所确定的而非单一厂商所制定，保证了其应用的通用性和跨平台性。同时，标准化使得 B / S 模式 可直接接入 Internet ，具有良好的扩展性、伸缩性，可从不同厂家选择设备和服务。 多个服务器可以有备份，多个服务运行在同一个或多个不同的服务器上，使体系结构的规模可以得到很好的提升。在实际中，数据管理层往往成为系统能力的瓶颈。

4.2 体系结构概述

通过 4.1 部分的分析，可以知道三层 B/S 体系结构具有许多传统 C/S 体系结构不具备的优点。因此该体育成绩管理系统选择基于三层 B/S 结构设计的开发。

表现层在最外面，展现管理系统的界面，用户可以通过表现层直接与系统交互，进行一系列的操作，包括查询成绩、录入成绩等等；中间是业务逻辑层，联系前端与后端；最内部是数据库访问层，用来进行数据库的相关操作。

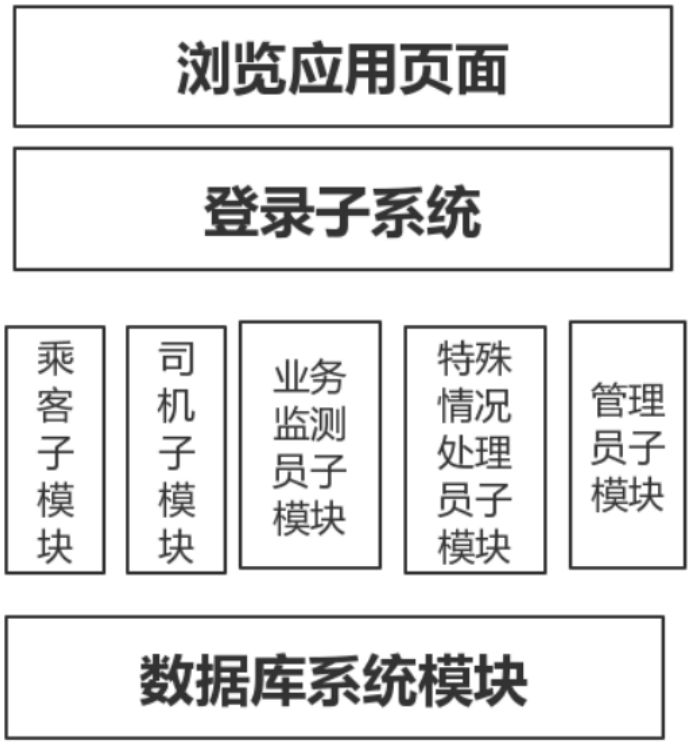
4.2.1 概念级体系结构

在概念级体系结构中，表现层负责向用户展示系统的页面及功能，业务逻辑层负责处理用户的请求并返回相应的数据，持久化层负责将用户的信息和数据保存在数据库中，并做必要的数据备份和恢复。

4-2 概念级体系结构示意图

4.2.2 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。模块级体系结构反应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。下图是按层次方法划分的软件系统结构。体系结构的层次表达了每层的上一层提供的功能和接口，以及需要使用的下一层的功能。



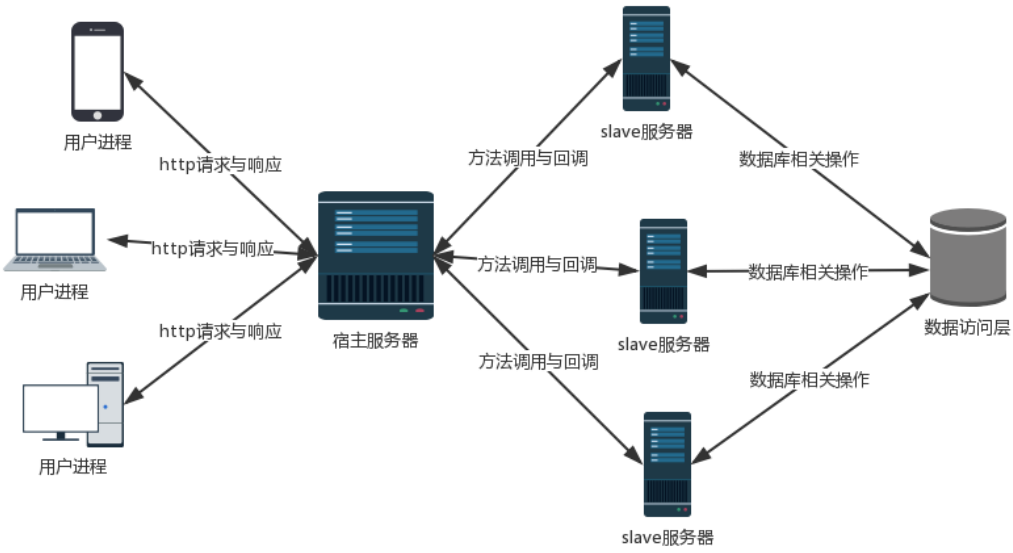
4-3 模块级体系结构示意图

4.2.3 运行级体系结构

运行级的体系结构用来描述系统的动态结构，描述上要尽可能用运行元素(例如，操作系统的任务、进程、地址空间等)、通信机制、运行元素分配的功能、以及资源分配等。同时，还要考虑到系统运行元素的位置分布、可移植性等因素。

- 每个用户都是一个独立的进程，可以独立地访问服务器发送请求并查看结果。

- 服务器为分布式架构,一台服务器作为Master服务器,管理Slave服务器的资源配置与负载均衡。
- 所有的 Slave 均访问同一个数据库，便于维护数据的一致性，同时定期对服务器进行备份，防止可能发送的故障和奔溃。

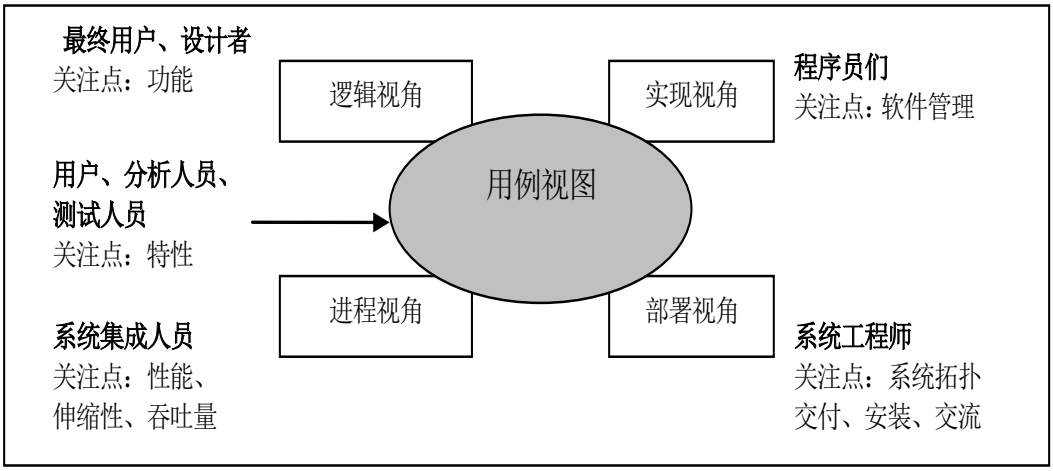


4-4 运行级体系结构示意图

4.3 结构化视图

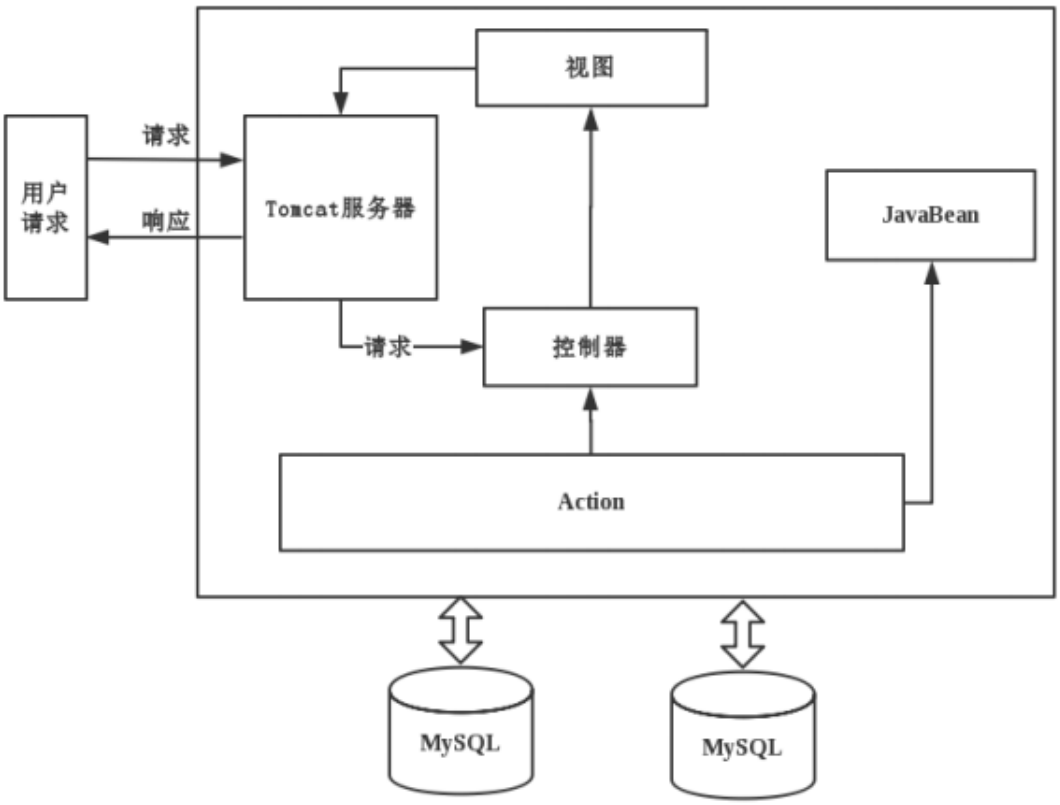
4.3.1 体系结构视角

Philippe Kruchten 提出从不同的角度勾画系统的蓝图,建立“4+1”视图模型。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观察，分析其影响系统的上下文和商业目标情况。

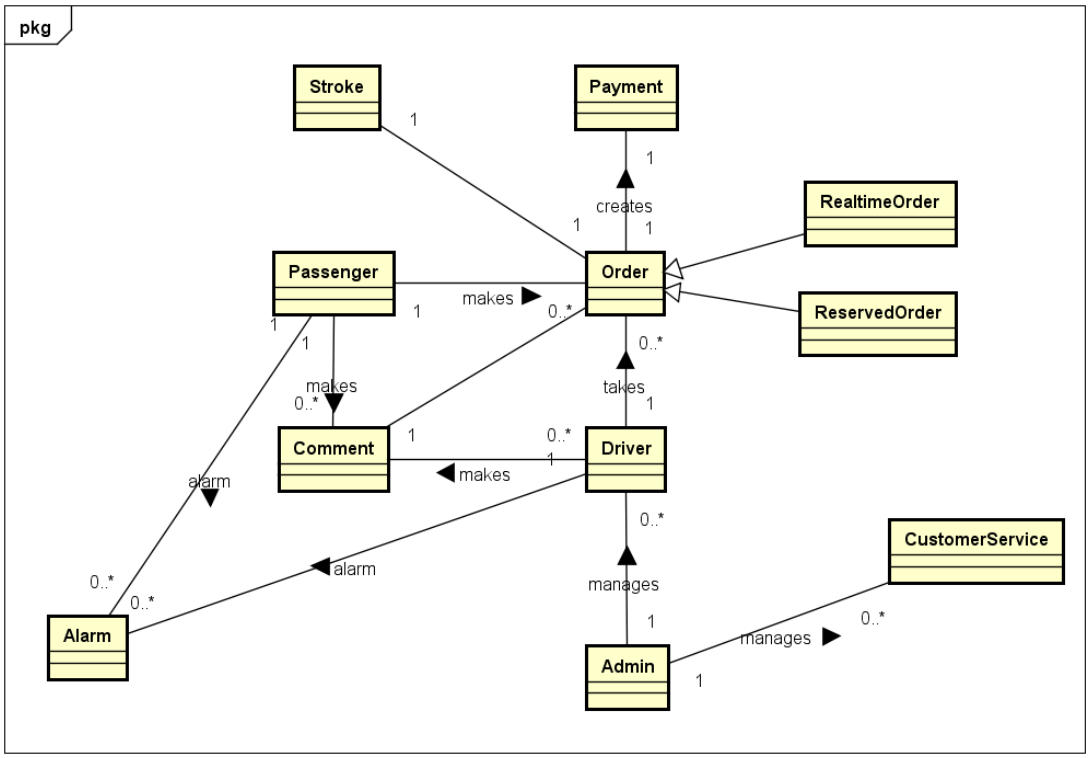


4-5“4+1”视图模型

4.3.2 逻辑视图



4-6 进程逻辑视图


pkg
Stroke
Payment
RealtimeOrder
Passenger
Order
ReservedOrder
Comment
Driver
CustomerService
Alarm
Admin
creates
makes
takes
alarm
manages
1
0..*

4-7 数据库逻辑视图

4.3.2 实现视角

实现视角，又叫开发视图（Development View），描述了在开发环境中软件的静态组 织结构，即关注软件开发环境下实际模块的组织，服务于软件编程人员。将软件打包成小 的程序块（程序库或子系统），它们可以由一位或几位开发人员来开发。子系统可以组织 成分层结构，每个层为上一层提供良好定义的接口。因此系统的架构开发视图如 3-11。

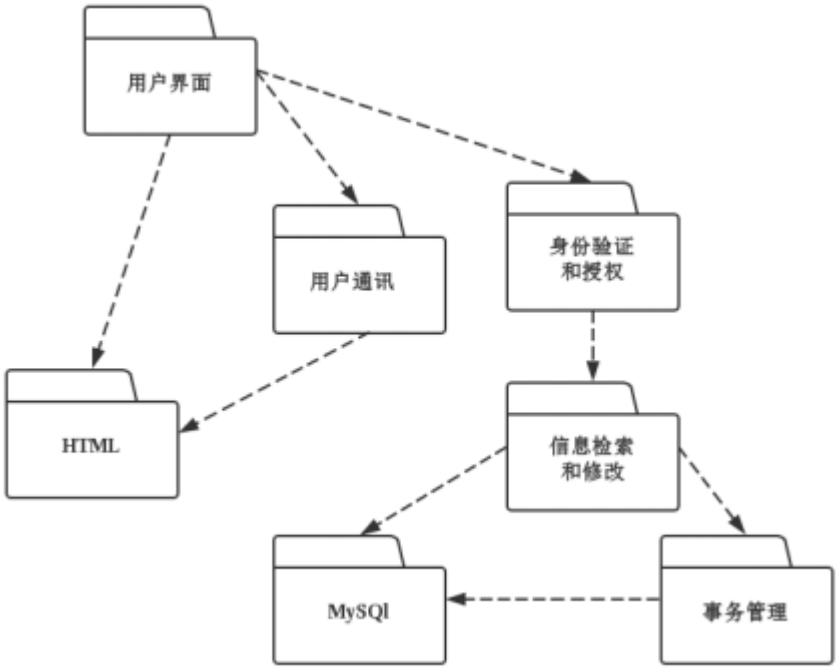


图 4-8 ． 架构开发视图

系统的实现视图用模块和子系统图来表达。完整的开发架构只有当所有软件元素被识 别后才能加以描述。但是，可以列出控制开发架构的规则：分块、分组和可见性。因此本 说明书设计系统功能图如 3-12

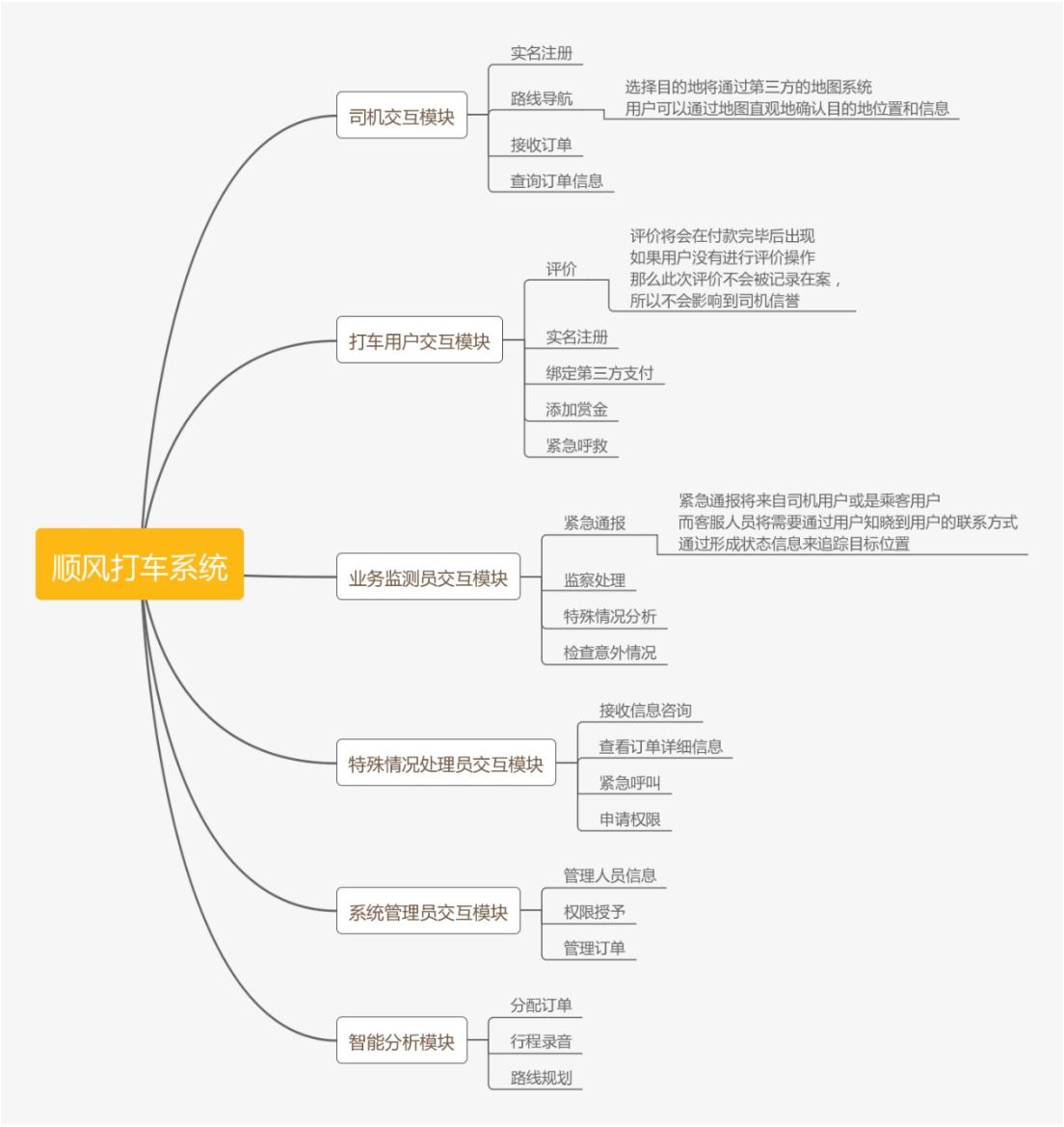


图 4-9 系统功能图

4.3.2 部署视角

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，本文设计部署视角如图 3-13.

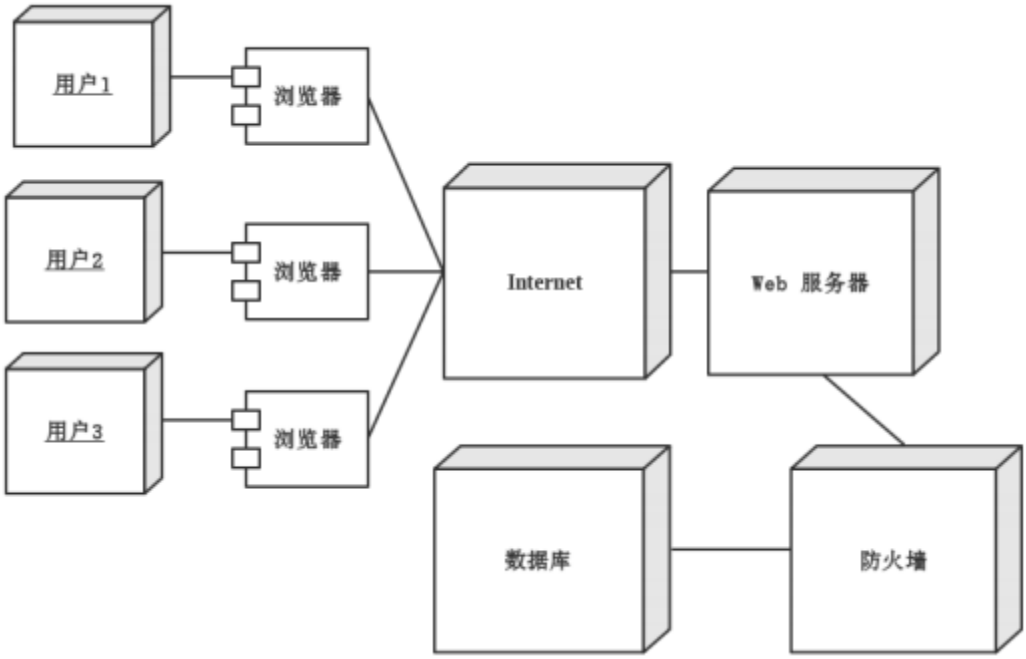


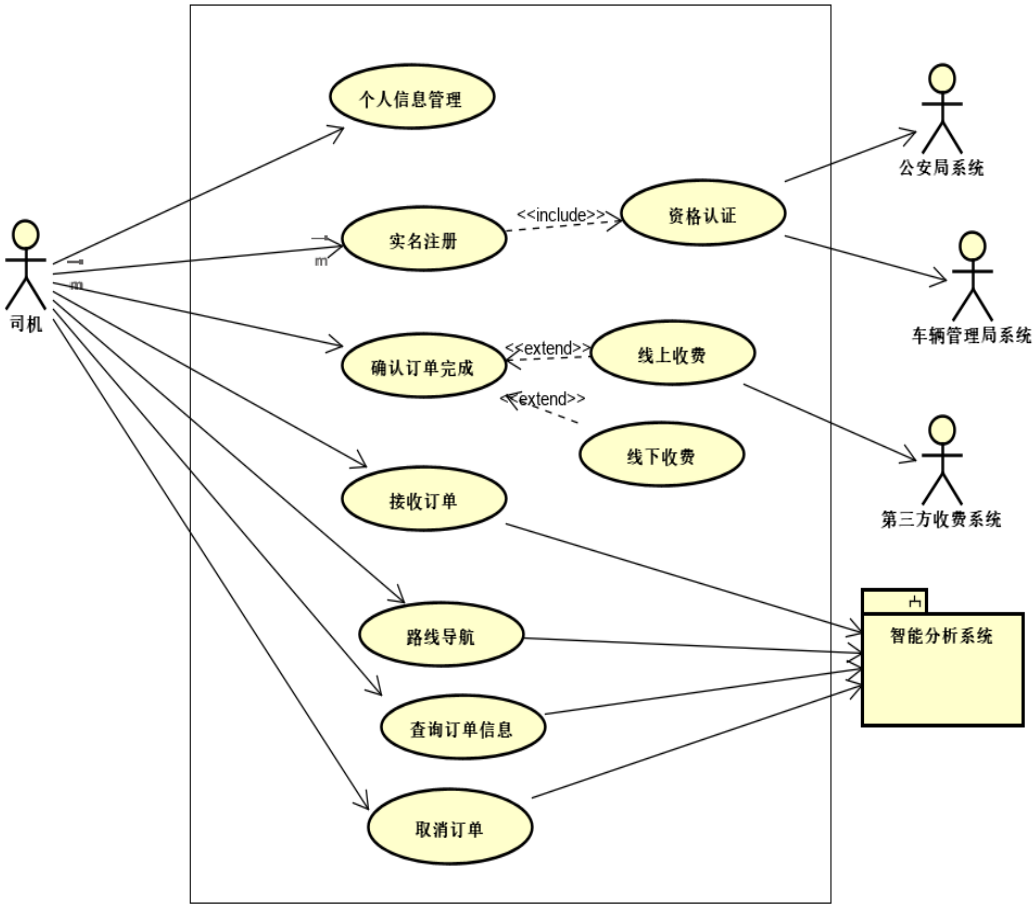
图 4-10 系统部署视图

4.3.2 用例视角

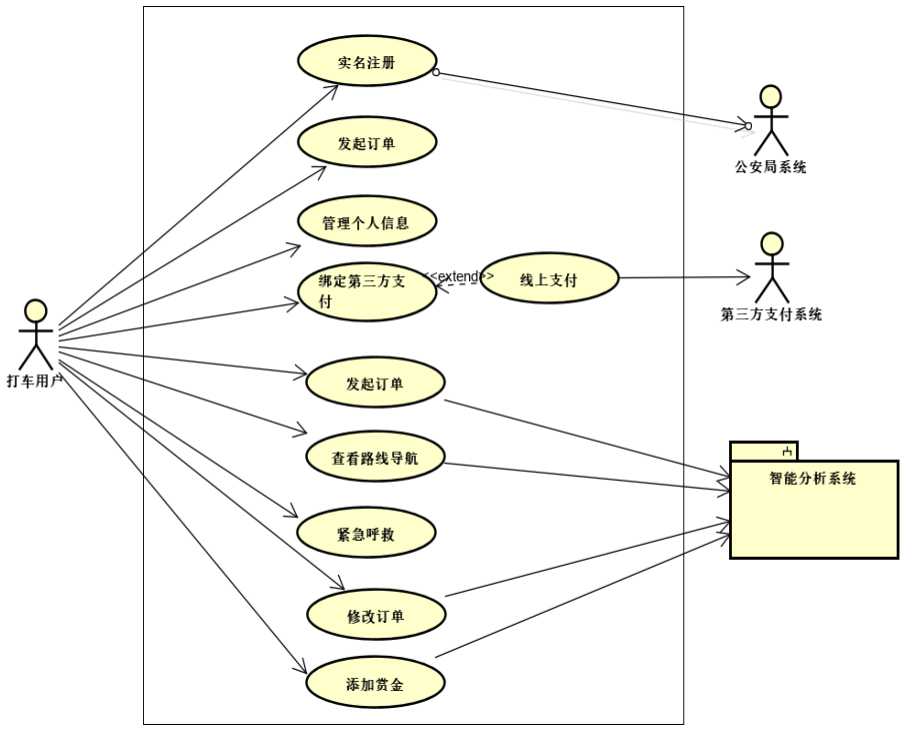
用例视角，又称“场景视图”，它综合所有的视图。用于刻画构件之间的相互关系，将四个视图有机地联系起来。可以描述一个特定的视图内的构件关系，也可以描述不同视图间的构件关系。需求报告中描述了一个版本的用例，在考虑体系结构时，又对其进行了一些修正，用例视角如下（用例文本详见 3.2 体系结构用例）：

用例图

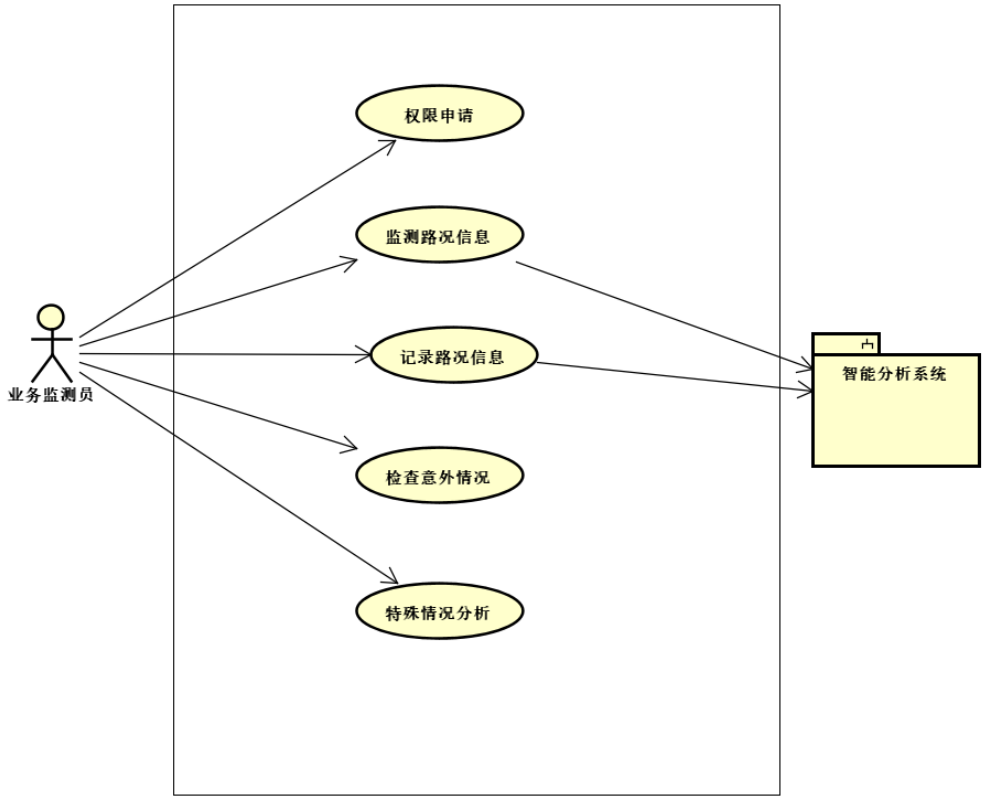
司机交互模块


个人信息管理
实名注册
<<include>>
资格认证
公安局系统
车辆管理局系统
司机
确认订单完成
<<extend>>
线上收费
extend>>
线下收费
第三方收费系统
接收订单
路线导航
智能分析系统
查询订单信息
取消订单

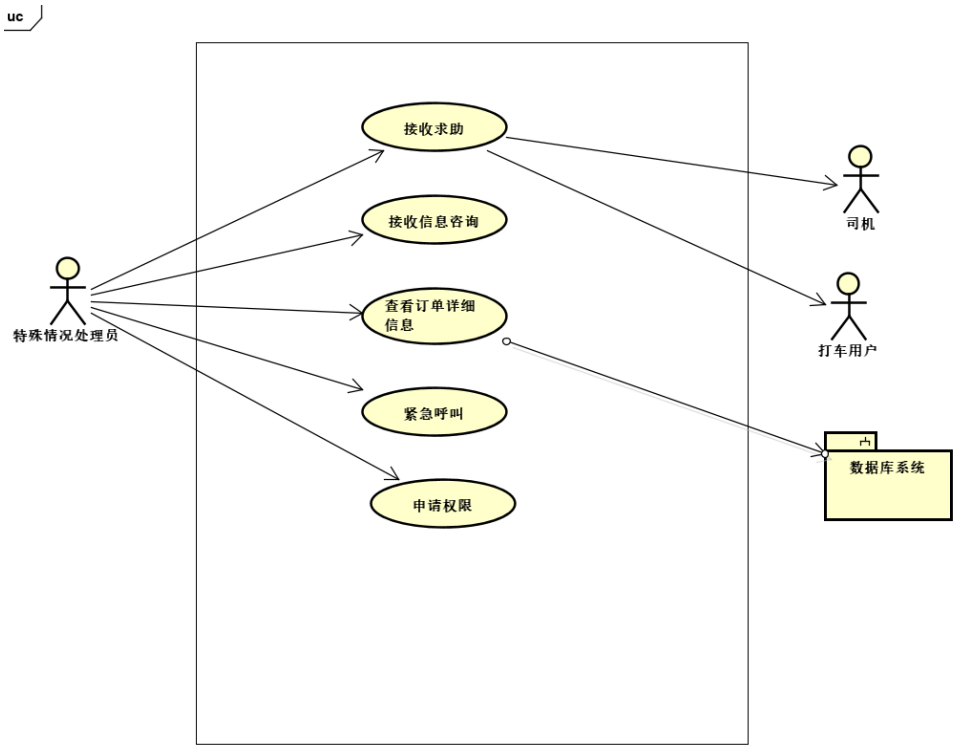
打车用户交互模块


实名注册
发起订单
公安局系统
管理个人信息
绑定第三方支付
<<extend>>
线上支付
第三方支付系统
打车用户
发起订单
查看路线导航
智能分析系统
紧急呼救
修改订单
添加赏金

业务监测员交互模块



特殊情况处理员交互模块



5.系统的质量分析与评价

5.1 场景分析

根据 ATAM 方法来对顺风打车系统的体系架构进行分析。

5.1.1 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：司机希望可以查看到自己的附近的停车位，并且可以看到自己的停车费用情况。该场景表明了用户期望系统易于使用，即易用性。

场景 2：管理员希望可以通过系统实时更新自己管理的车位信息，并且通过输入的车位计算方法自动计算出停车的费用，不再需要人工的参与。该场景表明了用户期望系统易于使用，即易用性。

场景 3：当管理员修改了车位信息后，系统要通知已经车位附近的司机用户，直到确认用户看到。该场景表明了用户期望系统易于使用，即易用性。

场景 4：用户在查询车位时，期望可以在 5s 内获得结果。该场景表明了系统的性能需求。

场景 5：当用户改变图形布局后，屏幕要在 1s 内画出来。该场景表明了系统的性能需求。

场景 6：当管理员在录入完车位信息后，期望系统可以自动生成一些统计信息和图表，以供自己查看车位状况。该场景表明了用户期望系统易于使用，即易用性。

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。

5.1.2 增长性场景

增长性场景 1：可增加新的数据库服务器、扩充现有的数据规模，降低远程用户的访问时间；

增长性场景 2：通过对软件系统适配手机界面，使得用户可以在移动

设备终端上访问顺风打车系统；

增长性场景 2：考虑使用分布式集群进行计算，均衡单个服务器的负载量，从而降低用户使用该软件系统的响应时间。

5.1.3 探索性场景

探索性场景 1：系统可以从 windows 平台切换到 Android 平台；

探索性场景 2：增加系统的可使用性，使其从 98%提升到 99%;

探索性场景 3：在正常情况下，当一半的服务器宕机时，可启动备用的服务器以致使用户感知不到服务器故障的发生，提升系统的可靠性；

1.优化算法，改进系统的性能，降低系统的响应时间；

2.提到系统对高并发访问的处理能力。

5.2 原型分析

使用效用树法进行原型分析，分析结果如下图所示

5-1 效用树分析图

5.3 风险

1. 开发费用预估不足，导致项目资金短缺，减缓了工程进度；

2. 需求分析不足，导致设计的体系结构并不满足项目的实际需求，在后期进行软件项目质量审核和评估时，进行大规模的返工。

3. 系统的重要构件、模块不符合设计的要求，导致系统功能的缺陷和项目的失败。

4. 系统的安全性设计存在漏洞，系统易受攻击且难以维护。

附录：

图示引用：