<打车软件系统>

体系结构设计文档

<4.0>

<2019.12.9>

<刘佳恒>

<学号：2017211972>
<班号：2017211504>

北京邮电大学软件学院

2019 年秋季学期

修订记录

日期	描述	作者	内容
2019-12-2	1.0	刘佳恒	体系结构文档框架设计
2019-12-3	2.0	刘佳恒	完善体系结构需求
2019-12-4	2.1	刘佳恒	体系结构模式设计
2019-12-5	2.2	刘佳恒	行为视图设计
2019-12-8	3.0	刘佳恒	文档框架调整
2019-12-9	4.0	刘佳恒	终稿

表格 1 修订记录

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
	刘佳恒		

表格 2 文档审批

目录

1.简介

在这一部分，主要介绍了本体系结构文档编写的目的、范围、概念定义、以及用到的参考文献。

1.1 目的

本文档为体系结构设计文档，旨在阐述打车软件系统的总体结构（逻辑设计、物理结构等），并设计好软件系统的总体设计策略以及所需要用到的技术，提高软件开发过程中的能见度。同时该文档还能使开发过程中涉及到的不同的人员方便地沟通和写作。使管理者便捷地对软件开发过程控制与管理，使得本系统的开发能够更加高效。

本体系结构设计文档作文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。

用户能够从本文档了解到软件系统的功能和整体结构；开发人员能够从本文档中了解到软件系统预期的功能以及相应的架构，并依此对系统进行设计和开发；测试人员能够从文档中了解到软件系统需要测试哪些部分；维护人员能够根据本文档知道如何对软件系统进行维护；而项目管理人员则能从本文档中预估软件系统开发的大致时间流程以及制定相应的开发计划

1.2 范围

（1） 软件系统名称：打车软件系统。

（2） 文档内容：本体系结构设计文档分析了打车软件系统的功能，该软件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质量和可靠性要求，并且分析了未来的升级维护中

可能遇到的问题及相应的解决方案。

（3） 应用范围：本软件体系结构设计文档适合于打车软件系统的总体应用结构，目的是满足系统的质量和可信赖性要求，以及打车软件系统未来的维护、运行和升级改造等要求。

1.3 定义、缩略语和缩写

名称	定义
打车软件系统	本文档负责的软件系统
打车用户（乘客）	使用该系统乘客用户，通过该系统实现个人信息查询修改、即时叫车、预约叫车、用车评价、寻求客服帮助等功能
司机	使用该系统的司机用户，通过该系统实现个人信息查询修改、接单载客、订单处理（执行订单）、路况信息获取等功能
业务监测员	使用该系统的业务监测员用户，通过该系统可以获取所有订单信息、路况信息，将信息处理并告知司机或乘客
用户需求	关于系统服务和约束的自然语言加上方块图表述。为客户撰写。
系统需求	一个结构化的文档写出系统的服务。作为客户和承包商之间的合同内容。
功能需求	系统需要提供的服务的表述，系统应该如何响应特定输入，系统在特定的 情形下应该如何动作
非功能需求	系统提供的服务或功能上的约束，例如时间约束、开发过程约束、标准等。

表格 3 定义与术语

缩写	名词解释
E-R 图	Entity Relationship Diagram 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。
UML	Unified Modeling Language 统一建模语言或标准建模语言 为软件开发的所有阶段提供模型化和可视化支持，由需求分析到规格，再到构造和配置。
SDL	Specification and Description Language 规格和描述性语言 描述一个现实世界中特定事件的发生过程的时序关系以及步骤。
DFD	Data Flow Diagrams 数据流图 数据流图从功能的角度对系统建模，追踪数据的处理有助于全面地理解系统，数据流图也可用于描述系统和外部系统之间的数据交换。
B/S	Browser/Server 浏览器/服务器
C/S	Client/Server 客户端/服务器
MVC	Model－view－controller 模型-视图-控制器 是软件工程中的一种软件架构模式，把软件系统分为三个基本部分：模型、视图和控制器。目的是实现一种动态的程序设计，使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。
Use Case	用例图 用例图是指由参与者、用例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。

表格 4 缩写与解释

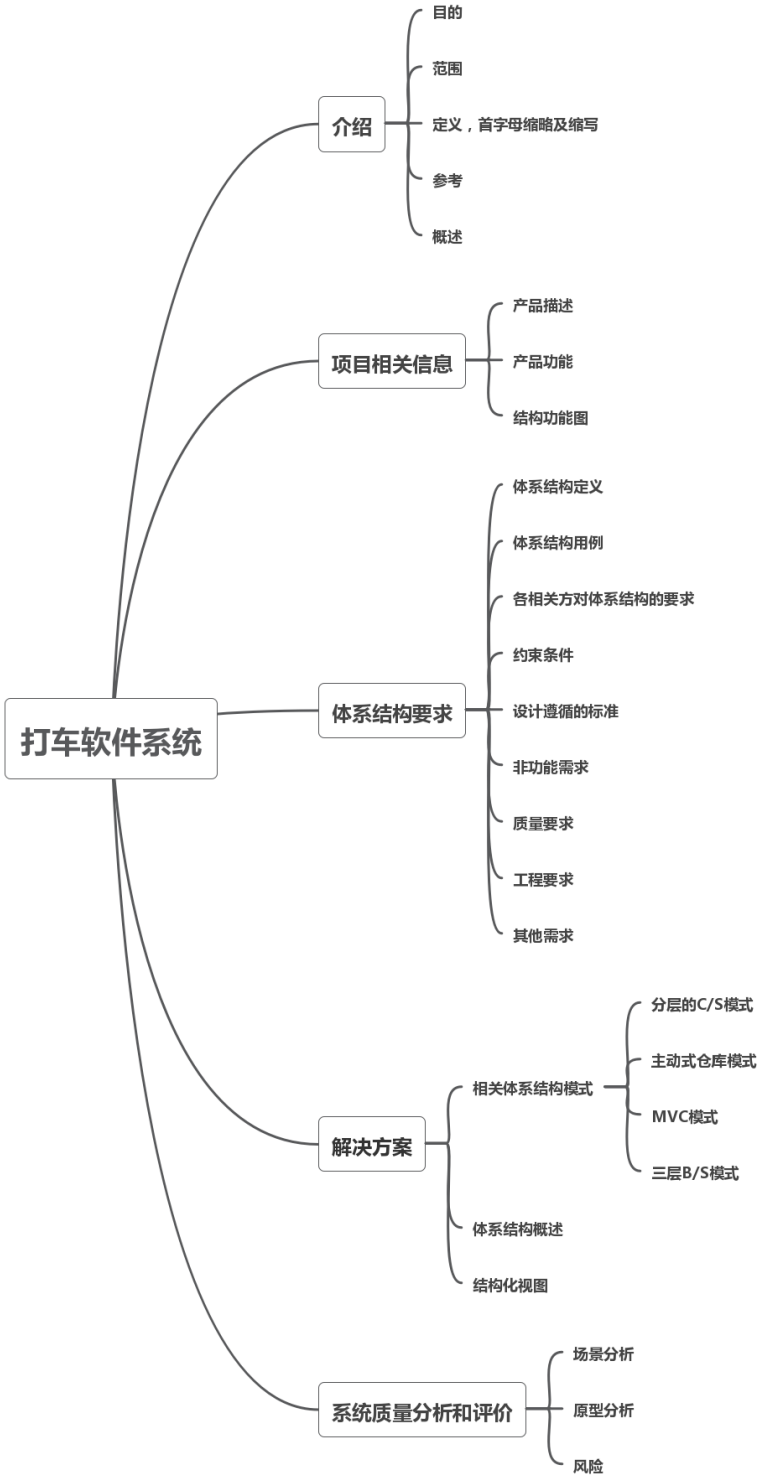
1.4 参考文献

[1] 王安生,《软件工程化》[M].北京:清华大学出版社，2014

1.5 概述

本文档将依次介绍：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。

即按照下图进行介绍：



图表 1 打车软件系统体系结构图

2. 项目相关信息

2.1 产品描述

随着“后 PC 时代”的到来，智能手机用户爆炸式的增长普及，移动互联网领域大有可为。城市化的快速发展，使得打车难的问题变得日益突出，给百姓的出行带来了诸多不便。首先，在出行高峰期，如上下班时，传统出租车数量不足，用户很难打到出租车，或是只能与他人拼车，影响用户的乘车体验；其次，由于传统出租车的运营方式不够人性化，也影响着用户的乘车体验。所以，打车软件应运而生，客观需求不断增长。

此打车软件系统旨在解决当前打车难、出租车体验差等情况，使得用户乘车体验更加良好，出行更加方便，收费内容一目了然。并且方便司机操作，功能齐全。

基本业务流程与市面内滴滴打车、高德打车、百度打车等打车软件相同。

该系统既可以完全独立，也可以作为其他打车系统的子系统。

2.2 用户角色

●打车用户分群

1. 学生群体：接受信息的方式更加多元化，容易接受新事物，所以学生更易尝试该打车软件系统，是系统的首批用户，但经济不宽裕，可以为其设计拼车功能。
2. 工作群体：因工作原因对打车的需求比较大，是系统的主要用户，但对打车的速度和效率要求比较高，可以为其设计优享车型。
3. 老人群体：不易学习、接受新兴事物，所以界面设计应当简洁易用，为其设计一键叫车及语音叫车功能。

●司机

1. 司机使用时间一般在固定开车时，为了交通安全，要求系统对语音读入友好，界面简单不繁琐，功能入口明细易操作。

2. 每天的用车高峰时段固定，此时会出现大量的司机同一时间访问系统的情况，这要求系统具有一定的并发性，使系统即使在并发访问量超过范围的情况下，仍能保持正常的使用。

●业务监测人员

1. 业务监测人员负责信息监听，寻找用车高峰时段、用车高峰地点，获取用户、司机信息，要求系统界面应当足够清晰，便于业务监测人员获取相应信息，得到想要的结果。
2. 业务监测人员的数量应当与城市规模有关，依照城市人口、面积设置相应的业务监测人员数量。

●系统维护人员

1. 由于系统的用户、司机信息会不断变更，如新用户注册、新司机注册等，需要有针对系统中用户基本信息数据的维护人员。
2. 由于每个月产生的订单信息数量巨大，需要维护人员定期进行数据库的备份，出错时可以进行错误恢复。

●系统开发人员

1. 系统开发人员在开发过程中，也可能出现偏差，需要根据需求分析文档进行不断调整，如修改、细化、添加相应的功能模块，以确保开发出来的系统满足客户方的要求和最终用户的实际需求。

2.3 产品功能

该打车软件系统能够及时的采集并为用户呈现附近可用的车辆，通过智能化和信息化的方式下单接单，可以实现的功能有：

打车用户：

1. 个人信息查询：查询个人信息，个人绑定手机号、绑定微信号、绑定支付宝等，能够查看历史订单与正在进行的订单。
2. 即时叫车：用户打车下单后，司机可以看到附近的叫车信息（出发地、目的地等），司机选择接单后，该用户收到接单通知，可以与司机进行交

流（电话、聊天窗口等形式），等待司机到来，完成即时叫车。用户可以选择打车类型，拼车、快车、优享车，不同类型的车收费不同。

3. 预约叫车：用户可以发布自己未来的乘车信息，包括出发时间、出发地点、目的地，发布信息后，司机可以看到所有的预约打车信息，选择合适的信息接单，此时打车用户可以与司机进行交流（电话、聊天窗口等形式），商定订单信息（修改、取消等），完成预约叫车功能。
4. 用车评价：用户在到达目的地完成该单时，可以对本次乘车进行评价。
5. 支付：用户可以选择扣款顺序，设置自己的扣款账号，如微信账号或支付宝账号。
6. 寻求客服帮助：用户可以直接与客服人员联系

司机：

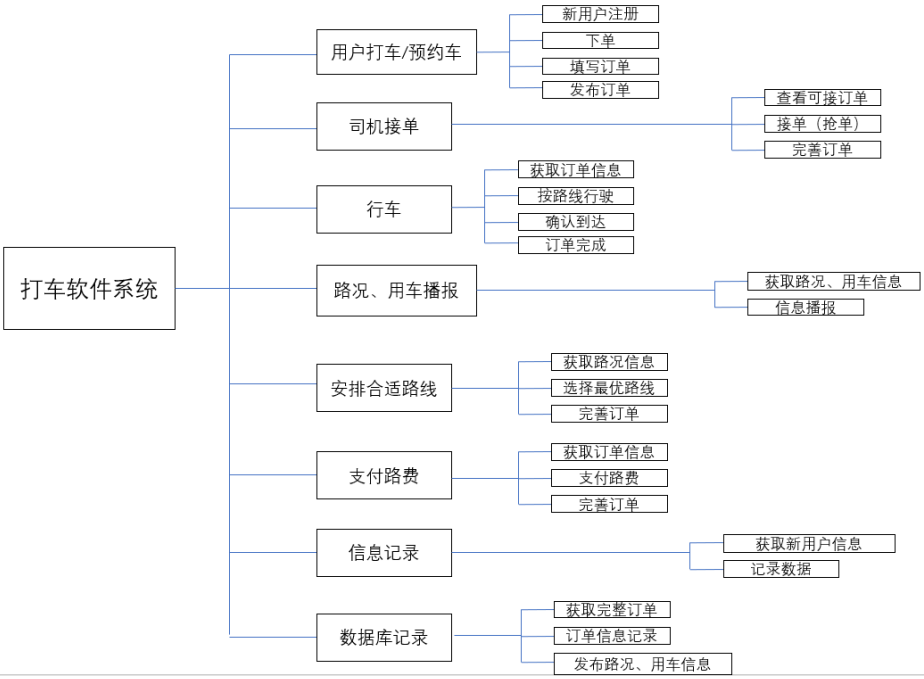
1. 接单载客：用户提交打车申请后，司机软件端会显示附近下单的用户信息（后续详细说明派车流程策略），包括出发地、目的地等。接单后可以与用户进行交流（电话、聊天窗口等形式）
2. 个人信息查询：司机可以看到自己正在进行的订单、已完成的订单以及接下的预约订单信息等。
3. 确认到达（订单完成）：当司机将打车用户送至目的地后，需点击确认到达，此时该订单完成，进入支付阶段。
4. 路况信息服务：司机可以查看实时路况信息，在地图上显示路况信息的同时可以语音提示重要路况信息，如某地点堵车建议绕行。

业务监测员：

1. 信息获取：业务监测员可以通过系统获得所有订单的发生信息。可以预测用车高峰时段、用车高峰地点，据此进行派车调度（可通过路况信息报告给司机）。

2.4 结构功能图

如上我们已经分析了产品的结构功能，下面用结构功能图来展示如上功能之间的关系。



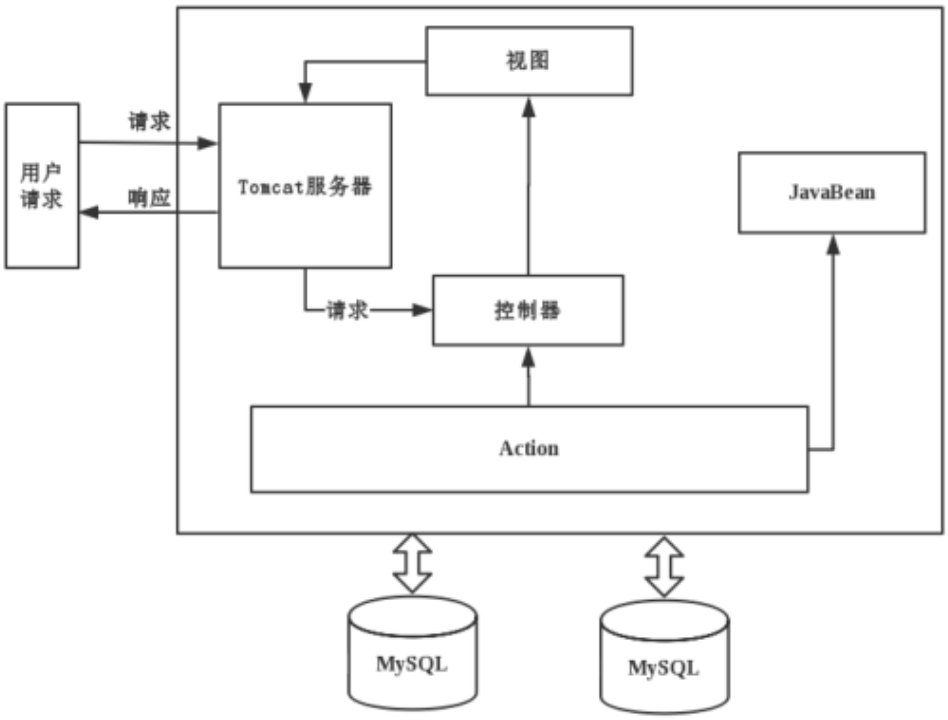
图表 2 “打车软件系统”的功能结构图

3. 架构表现

3.1 逻辑视角

本系统在软件逻辑架构的设计工作上采用的是轻量级 Struts 框架，该框架是基于 MVC 模式的一种架构。下图为本文系统的软件逻辑架构示意图

在 Sturst 框架技术中，Action 和 Actionservlet 具有控制器的功能，而 JavaBean 则是用来充当系统的控制模型的角色。具体实现的过程为:当用户提出操作请求之后，第一步会由 Tomcat 服务器负责接收用户发出的操作请求的信息，在将此信息请求传送到控制器 Actionservlet 中，接下来由 Action 来负责接收操作指令，然后从数据库 MySQL 中找到的对应信息(如车位信息、路线规划信息、司机相关信息、打车者相关信息等)，之后再将数据访问结果返回给视图呈现给用户。

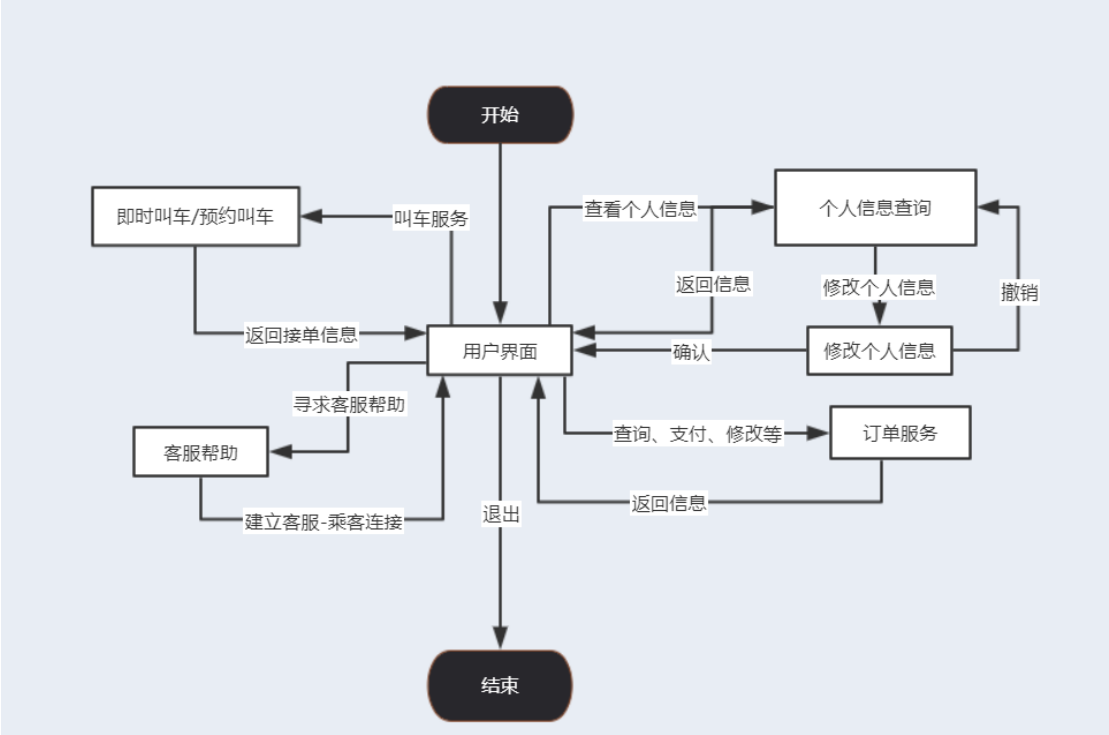


图表 3 逻辑视角图

3.2 进程视角

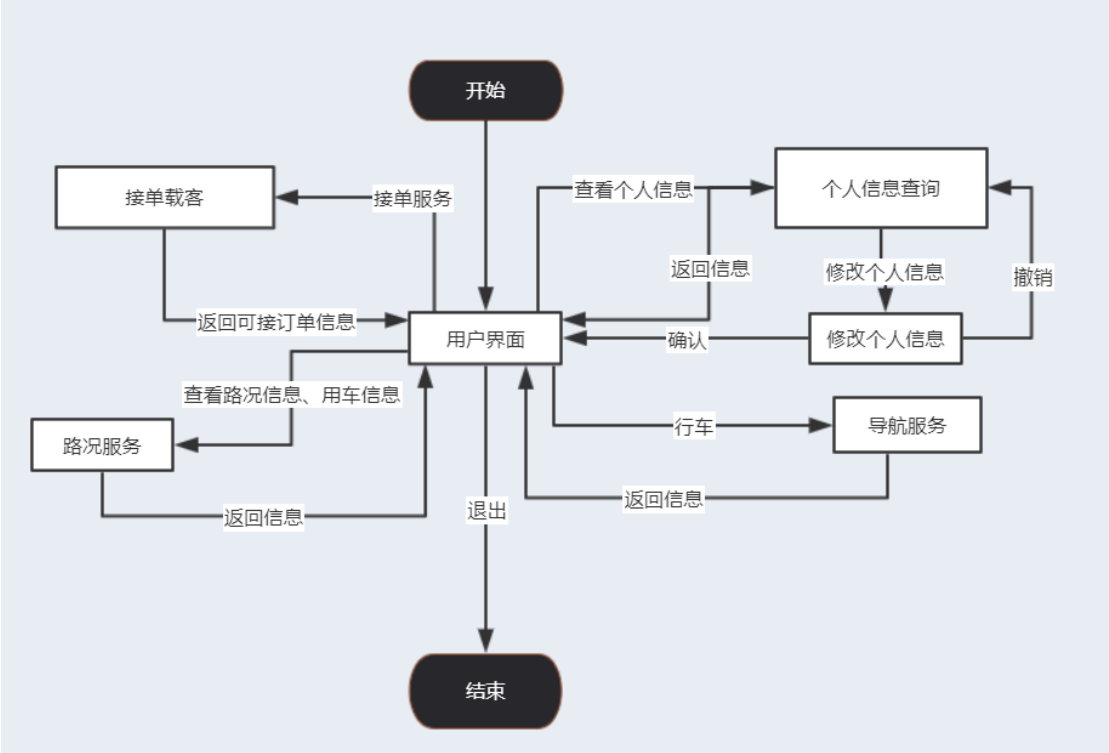
本说明书使用 SDL 语言对针对进程视角进行描述。主要描述在概念级的软件体系结构下，系统运行态的情况。描述系统在执行时，包括哪些进程（包括线程、进程、进程组），以及它们之间是如何进行通信的、如何进行消息传递、接口如何。并且来说明如何进行组织。。本打车软件系统根据用户的不同需求授予用户不同的权限来满足用户不同的功能需求，不同用户对于系统可执行的操作也不同。本系统的用户主要为三类：打车用户（乘客）、司机、业务监测员。为了更加清晰地描述本系统的用户行为与程序功能模块的对应关系，下面结合不同的用户和其对应的功能，分别给出用户的行为视图。

3.2.1 打车用户（乘客）行为视图



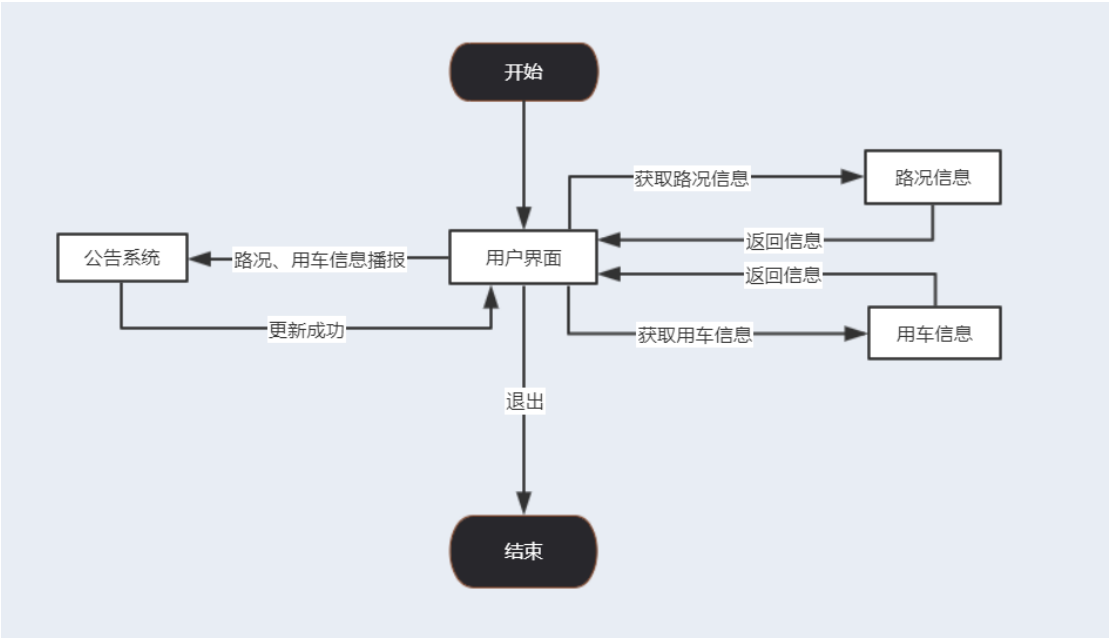
图表 4 打车用户（乘客）行为视图

3.2.2 司机行为视图



图表 5 司机行为视图

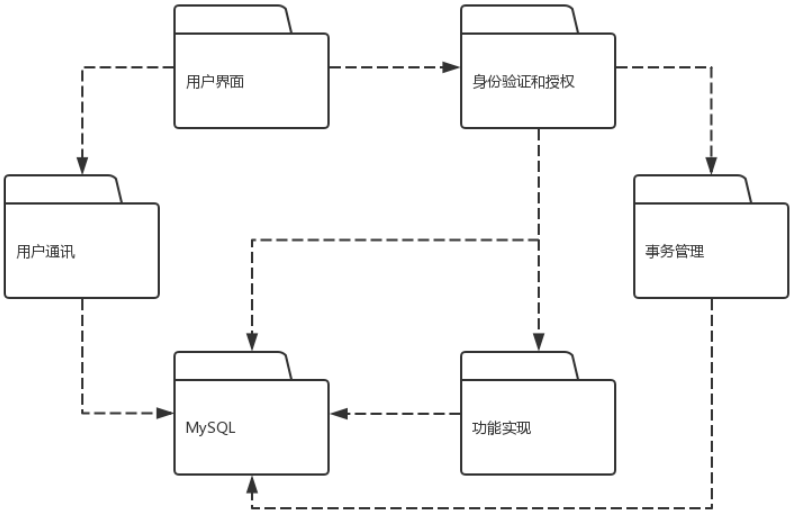
3.2.3 业务监测员行为视图



图表 6 业务监测员行为视图

3.3 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。



图表 7 实现视角图

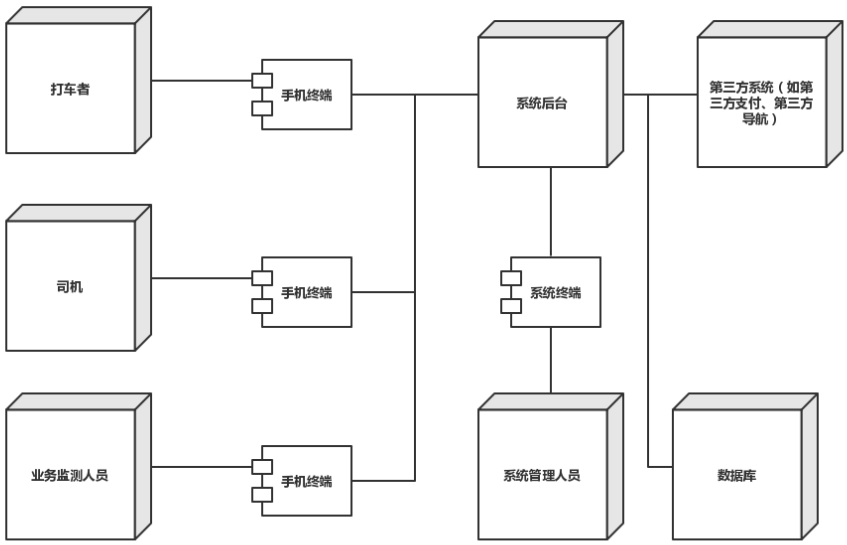
1. 软硬件条件的限制。考虑到本系统模块繁杂，要求严格，服务覆盖范围广，在线人数大，对于软件的开发来说都是很大的挑战。测试如此复杂的分布式系统需要耗费巨大的硬件资源，这对于项目提供的软硬件条件来说都是挑战。
2. 软件系统设计人员对系统理解不到位。本打车软件系统同时采用了三种设计模式交叉使用，不同的层次交叉并行，任何一个小的疏忽都可能导致工程最终的失败。要求软件系统设计人员对于系统整体架构有着清晰的把控。
3. 软件开发人员能力限制。多种架构模式交叉使用，面向多种用户的客户端并行开发，快速准确的数据库更新，用户信息的安全保障，同时应对这些开发任务对于软件开发人员与系统架构师来说是很大的挑战。
4. 时间限制。按照体系结构文档的要求，第一版系统需要在六个月交付，时间短，开发压力大，对于系统的实现是一个很大的问题。

   缺少经验。对于面向政府的大型分布式系统，企业可能之前并没有相关的开

发经验，对于工程流程管控，项目预算，服务质量保障等可能缺少相关的经验。

3.4 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下图：



图表 8 部署视角图

4. 架构目标和约束

4.1 架构目标

本文档在原有需求分析基础上，进一步考虑，试图设计一个通用的体系结构，能够支持滴滴打车、高德打车、百度打车等软件的开发。

4.2 设计约束

4.2.1 实用性原则

在打车软件系统的设计过程中，要充分考虑各个公司（如滴滴、高德、百度）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，效率优异，能够良好的运行。切忌纸上谈兵，看似用了一些很炫酷的框架实则不具备良好的实用效果。

4.2.2 稳定性原则

打车软件系统提供的 24×7 小时不间断服务，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的服务中断的原因。给用户提供稳定的服务。

4.2.3 复用性原则

组件复用是打车软件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公司开发出适合他们实际情况的打车软件系统。

4.2.4 扩展性原则

打车软件系统必须具有良好地可扩展性。随着移动互联设备的兴起，越来越多的用户选择在终端上访问互联网，因此，该打车软件系统必须在未来扩展到能够适配移动终端的访问。以满足用户的需求。打车软件系统必须能顺应这种业务的急剧扩展而快速进行相关的扩展开发。

4.2.5 安全性原则

打车软件系统带来了业务的扩展性，应用的延伸性、用户的方便性等优势的同时，也带来了风险的集中。打车软件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。

4.3 逻辑数据库要求

要求使用关系型数据库来实现数据的存储与管理。由于 SQL 是开源免费的，所以从节约成本角度要求必须支持基于关系型数据库 SQL Server，同时从未来可扩展的角度上也要求其支持 DB2。

4.4 其他约束

1）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。

2）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：Win7，win8,win10。

3）用微信公众号及小程序实现 “打车软件系统”。

4）合法申请微信公众号及小程序的开发使用权限。

5）小程序启动后，有合理的用户界面，能实现人机交互的功能。

4.4.1 各相关方对体系结构的要求

4.4.1.1 系统开发、测试、维护人员

系统应具备高可维护性、高可移植性，以便于软件系统的复用和开发维护

可维护性包括：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。

可移植性包括：系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。

4.4.1.2 用户

系统应具备高可用性。

高可用性包括：方便操作，具备一定的容错能力并且能够对用户使用过程中的才做进行规范的提示。

4.4.2 非功能需求

4.4.2.1. 工程要求

设计要素		主要约束
运行环境	操作系统	Windows 10/Linux 10 及以上
	数据库	MySQL 14.0 及以上
	微信小程序/微信公众号	Android/IOS
用户端 PC 软件	操作系统	Windows/OSX/Linux
	浏览器	Chrome/Edge/Firefox/Safari
开发环境支持	操作系统	Linux Ubuntu 16.04
	开发工具	eclipse
	Web 服务器	WebLogic
	CPU	3.4 GHz Intel Core i7
	内存	16GB

表格 5 工程要求

4.4.2.2 其他需求

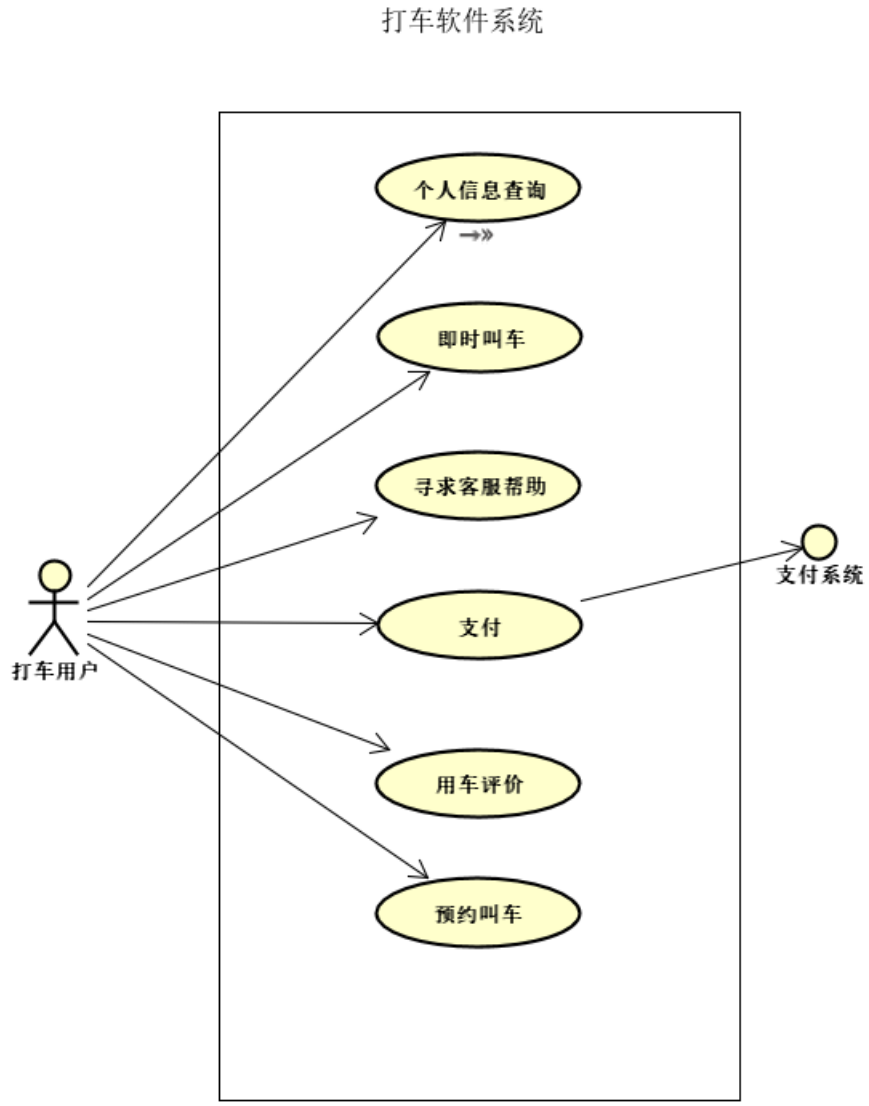
界面需求：界面应有很好的交互体验，便于操作，对于没有计算机基础的打车用

户（乘客）、司机来说可以很容易理解系统的各个功能。

数据容量的需求：对于每个季度结束后的订单数据可以进行合并记录，留下汇总型数据，使得数据库有足够容量去处理新一季度的订单信息。

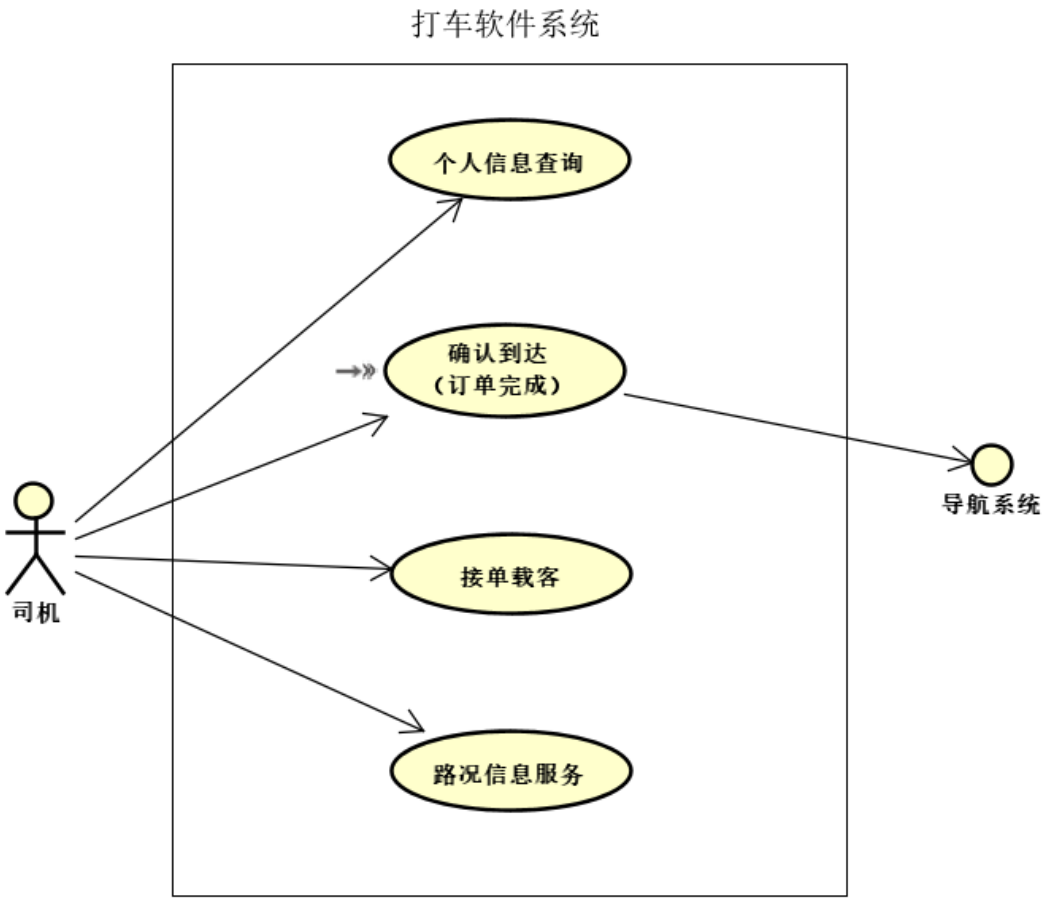
5.用例视图

5.1 用例图-1 打车用户



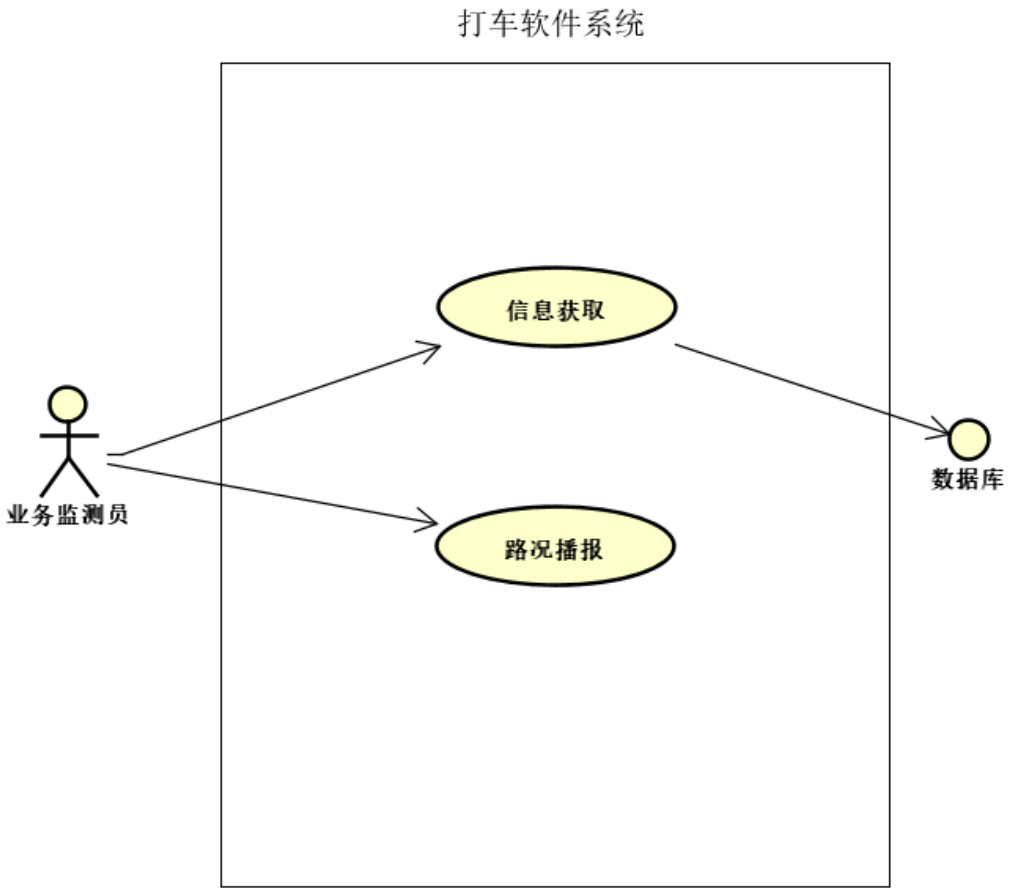
图表 9 用例图-1 打车用户

5.2 用例图-2 司机



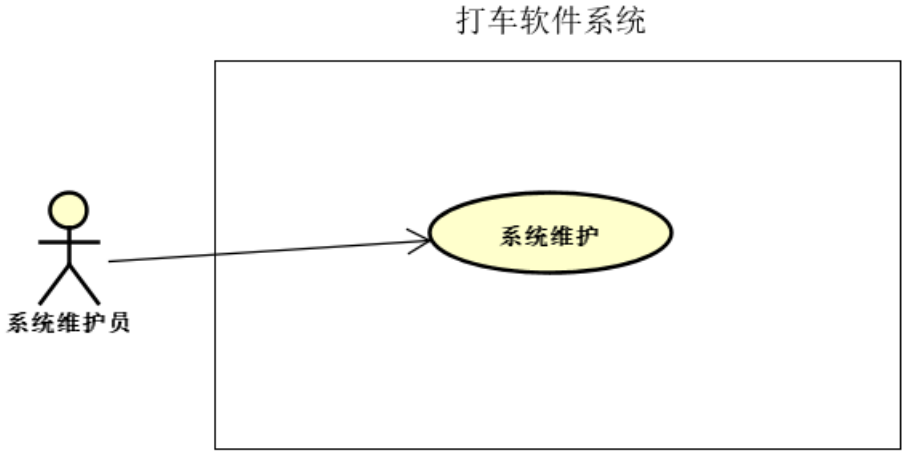
图表 10 用例图-2 司机

5.3 用例图-3 业务监测员



图表 11 用例图-3 业务监测员

5.4 用例图-4 系统维护员



图表 12 用例图-4 系统维护员

6. 逻辑视图

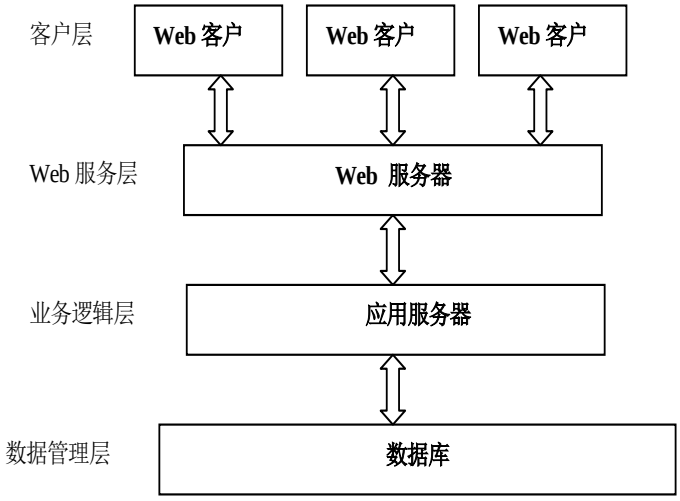
6.1 体系结构概述----相关的体系结构模式

6.1.1 分层的 C/S 模式

1. 分割：表现层、业务层和数据处理逻辑被分割到不同的层面，结构清晰。
2. 在我们的使用场景下，司机大多使用移动客户端进行系统登陆、查询与下单，使得 B/S 架构不能很好的满足用户的需求。在这种场景下，开发方便，操作简单，层次清晰的 C/S 架构必然是首选。
3. 各层之间异步通信：层与层之间的通信是异步的“请求-响应(request-reply)”。请求是单方向的，从客户层开始，经过 Web 和业务逻辑层，再到数据管理层。每层都等待其它层的处理响应。
4. 部署灵活：分层结构不限制多层应用的部署方式。所有层可以运行在一台机器上，或者，每层部署到一台独立的机器上。

5. 标准化：分层可以实现各层的标准化，例如，在 Web 应用中，客户端可以使用不同厂家的浏览器，兼容地对不同厂家 Web 服务器访问。

采用分层分布式系统框架的 C/S 结构示意图：



图表 13 分层分布式系统框架的 C/S 结构示意图

6.1.2 主动仓库模式

在需要数据被多个部件共享的情况下，共享仓库模式给予了用户很好的机制和选择。相比于顺序体系结构，其优点在于实现了数据共享，以及不相邻的两个部件之间的信息交流。在我们的系统中，数据经常是被动的访问，并经常发生更新。在这种情况下，如果仓库中的数据改变或数据访问发生变化，客户端必须立刻得到通知。

主动仓库模式是共享仓库模式化的一个变体，要求把仓库中发生的特殊事件主动地通知到客户端。主动式仓库要维护客户端的注册表，并通过适当的提示机制通知客户端。

6.1.3 MVC 模式

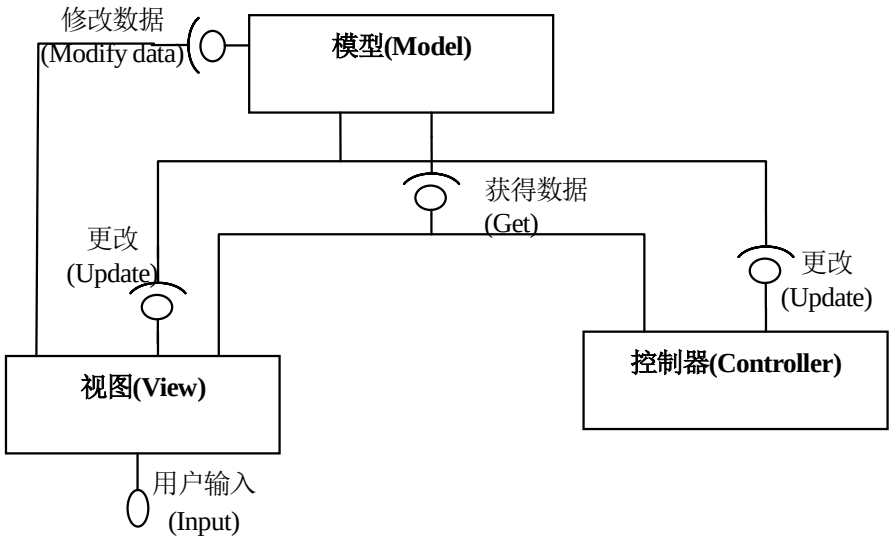
一个系统往往需要提供多个用户界面，而每个用户界面可能只需要反应一部分应用数据。当这些数据改变时，要能够自动和灵活地反应到不同的用户界面上。

这就需要能够很容易地修改其中的一些用户界面，而不需要修改与数据相关联的应用逻辑。

解决的方法是将系统划分为三个部分：

1. 一个模型(Model)：封装应用数据及其对这些数据的操作，与用户界面独立开；
2. 一个或多个视图(Views)：向用户展示指定的数据；
3. 一个控制器(Controller)：与每个视图关联起来，接收用户的输入，并将它翻译成对 Model 的请求。

View 和 Controller 组成了用户界面。依据 MVC 模型，更改所有 View 和 Controller 的通知机制可以用“发布-订阅(Publish- Subscribe)”模式实现。所有控制器和视图从模型接收到的订阅都要发布出通知。用户只通过 View 及其 Controller 进行交互，而不依赖于 Model，反过来，将更改情况通知给所有的不同用户。



图表 14 MVC 模型图

6.1.4 三层 B/S 模式

1. 开放的标准

B/S 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，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.较低的开发和维护成本

B/S 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。

3.使用简单，界面友好

B/S 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

B/S 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

B/S 系统在客户机与数据库 服务器之间增加了一层 Web 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

B/S 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

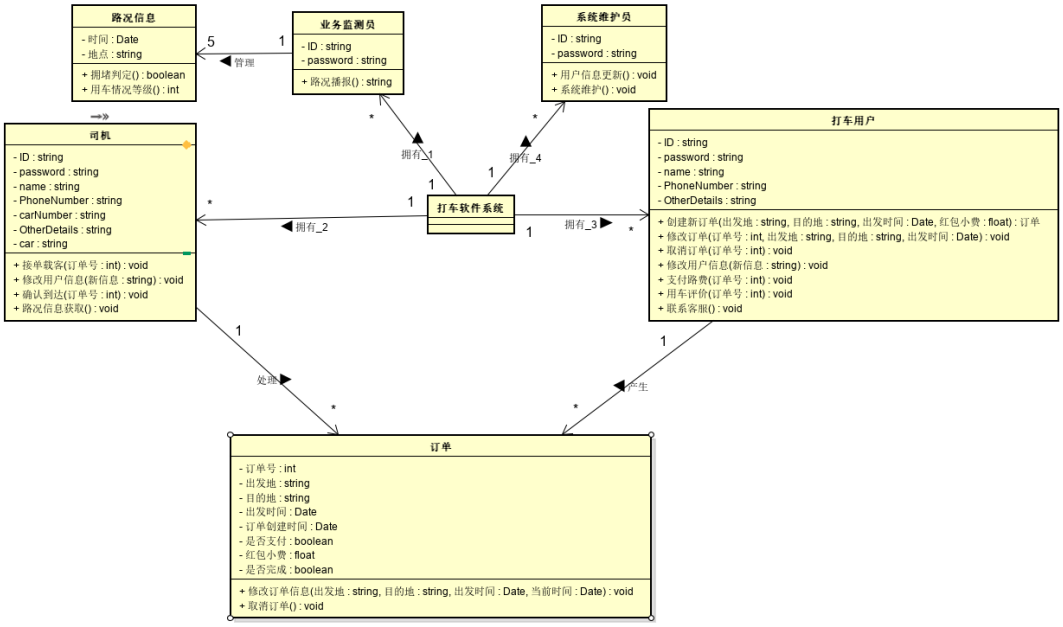
Client / Server 系统是基于局域网的，而 B/S 系统无论是 PSTN、 DDN、帧中继， X.25 、 ISDN，还是新出现的 CATV、 ADSL，B/S 结构均能透明的使用。

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	B rowser / 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	B rowser / 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	B rowser / 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。

表格 6 三层 B/S 体系结构对质量属性的影响

6.2 体系结构概述----类图关系



图表 15 类图关系

7.流程视图

7.1 流程

7.1.1 用户个人信息查询

7.1.1.1 介绍

用户在登录后，可以查询个人信息，个人绑定手机号、绑定微信号、绑定支付宝等，能够查看历史订单与正在进行的订单。

7.1.1.2 输入

用户点击“我的-个人信息”这一选项即可。

7.1.1.3 过程

用户点击“我的-个人信息”，系统呈现出个人基本信息，包括用户昵称、个人绑定手机号、绑定微信号、绑定支付宝、订单信息（正在进行的订单、已完成订单）。

7.1.1.4 输出

显示出用户关注的信息。

7.1.1.5 错误处理

当用户未登录时，系统提示用户：请先登录。自动跳转至登录/注册界面。

7.1.2 用户即时叫车

7.1.2.1 介绍

用户打车下单、填写订单信息后，司机可以看到附近的叫车信息（出发地、目的地等），司机选择接单后，该用户收到接单通知，等待司机带来，完成即时叫车。用户可以选择打车类型，拼车、快车、优享车，不同类型的车收费不同。

7.1.2.2 输入

用户选择打车，输入出发地、目的地，并选择出发时间、打车类型。

7.1.2.3 过程

用户输入完成后，点击“一键叫车”，等待司机接单。司机看到附近可接订单后，选择相应的订单接单，用户方可以看到接单通知，此时可以与司机进行交流（电话、聊天窗口等形式），商定订单信息（修改、取消等），等待司机到来，

即时叫车完成。

7.1.2.4 输出

可显示司机正在路上，能实时看到司机的位置。

7.1.2.5 错误处理

当用户未登录时，系统提示用户：请先登录。自动跳转至登录/注册界面。

用户输入错误的地点时，提示无效地点；当订单长时间未被接单，系统提示用户：附近可用车辆较少。

7.1.3 用户预约叫车

7.1.3.1 介绍

用户可以发布自己在未来时段的乘车信息，完成预约叫车功能 。

7.1.3.2 输入

用户填写自己未来的乘车信息，包括出发时间、出发地点、目的地。

7.1.3.3 过程

用户可以发布自己未来的乘车信息，包括出发时间、出发地点、目的地，发布信息后，司机可以看到所有的预约打车信息，选择合适的信息接单，此时打车用户可以与司机进行交流（电话、聊天窗口等形式），商定订单信息（修改、取消等）。预约叫车功能完成。

7.1.3.4 输出

用户点击“预约下单之后”，一直向用户呈现订单当前信息，是否被接单、接单司机信息等内容。当司机接单后，用户可以与司机进行交流（电话、聊天窗

口等形式），商定订单信息（修改、取消等）。

7.1.3.5 错误处理

当用户未登录时，系统提示用户：请先登录。自动跳转至登录/注册界面。

用户输入错误的地点时，提示无效地点；当订单未被接单直到用户预定的用车时间，系统提示用户：无合适车辆，请重新预约。

7.1.4 用户用车评价

7.1.4.1 介绍

订单完成、用户支付路费后，用户可以对本次用车进行评价。

7.1.4.2 输入

订单完成、用户支付路费后，系统显示本次订单详情页面，用户点击“评价”

7.1.4.3 过程

到达目的地后，司机确认到达，用户支付路费，系统显示本次订单详情页面，用户点击“评价”，即可以对本次用车进行评价打分（文字、图片等）。

7.1.4.4 输出

用户完成评价后，系统提示：评价成功。

7.1.4.5 错误处理

当用户未登录时，系统提示用户：请先登录。自动跳转至登录/注册界面。

当订单尚未完成、用户尚未支付路费时，不向用户提供评价途径。

7.1.5 用户支付车费

7.1.5.1 介绍

司机确认到达后，用户需要支付相应车费。

7.1.5.2 输入

用户可以选择“微信支付”、“支付宝支付”、“余额支付”、“银行卡支付”。

7.1.5.3 过程

司机确认到达后，用户的打车软件系统显示本次路程收费详情，用户点击“立即支付“，系统跳转至支付页面，用户选择“微信支付”、“支付宝支付”、“余额支付”、“银行卡支付”。此处实现与外部服务的接口。

7.1.5.4 输出

用户支付路费后，系统提示：支付成功。

7.1.5.5 错误处理

当用户未登录时，系统提示用户：请先登录。自动跳转至登录/注册界面。

当用户选择的支付方式对应的账号中余额不足时，系统提示：余额不足，支付失败。系统返回支付界面。

7.1.6 用户寻求客服帮助

7.1.6.1 介绍

当用户想要客服介入解决问题时，需要有与客服沟通的途径。

7.1.6.2 输入

用户可以点击“联系客服”。

7.1.6.3 过程

在打车软件界面有“联系客服”悬窗，用户可以点击“联系客服”，与客服取得联系。

7.1.6.4 输出

显示用户与客服联系聊天窗口，用户可以选择人工服务。

7.1.6.5 错误处理

当用户未登录时，系统提示用户：请先登录。自动跳转至登录/注册界面。
当前人工客服忙时，系统提示用户：当前人工客服较忙，请稍后再试。

7.1.7 司机接单载客（包括派车流程策略）

7.1.7.1 介绍

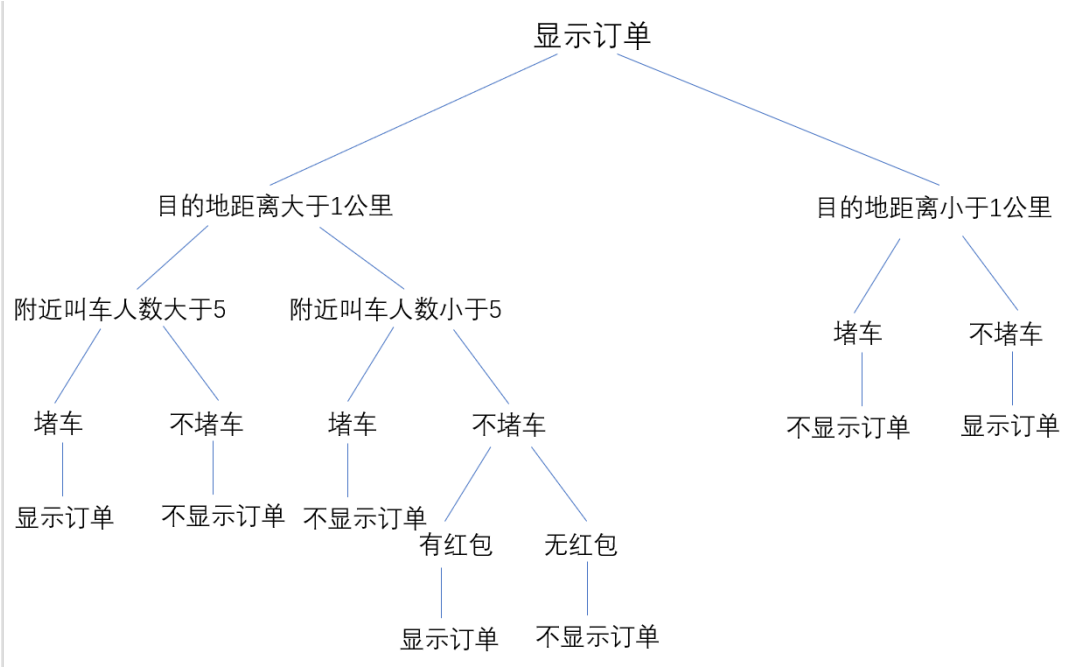
用户提交打车申请后，司机软件端会显示附近下单的用户信息，包括出发地、目的地等，司机能够进行接单。

7.1.7.2 输入

司机点击接单，即可完成接单。

7.1.7.3 过程

派车策略：



图表 16 派车策略图

首先根据派车策略，为司机显示合适的订单，司机可以在这些已经筛选出来的订单中选择订单接单。

7.1.7.4 输出

司机接单后，在司机的软件界面显示该订单的信息，在用户的软件界面显示已被接单的通知，并显示订单信息。

7.1.7.5 错误处理

当司机接单时，该订单可能已经被其他司机接单，此时向司机提示：该订单已被其他司机接单。

7.1.8 确认到达

7.1.8.1 介绍

到达目的地后，需要司机确认到达信息。

7.1.8.2 输入

司机点击“确认到达”。

7.1.8.3 过程

当司机将打车用户送至目的地后，点击“确认到达”。

7.1.8.4 输出

在司机系统界面显示订单完成，在用户系统界面显示支付界面。

7.1.8.5 错误处理

若司机尚未到达目的地附近点击“确认到达”，系统提示：尚未到达目的地，是否确认到达？若司机继续确认，可能是与打车用户协商修改了目的地，则确认到达；若司机取消，则继续行驶。

7.1.9 路况信息服务

7.1.9.1 介绍

司机能够看到实时路况信息，用于避开拥堵地点，或是前往用车人数较多的地区接单。

7.1.9.2 输入

无输入，在首界面即显示路况信息，用车情况等内容。

7.1.9.3 过程

司机在首页即可看到路况信息，点击可以查看详情。同时司机可以查看当前时段用车高峰地点，可以查看高峰时段等内容。

7.1.9.4 输出

显示司机关注的信息。

7.1.10 业务监测人员信息获取

7.1.10.1 介绍

业务监测员可以通过系统获得所有订单的发生信息。可以预测用车高峰时段、用车高峰地点，据此进行派车调度（可通过路况信息报告给司机）。

7.1.10.2 输入

业务监测人员无需输入。

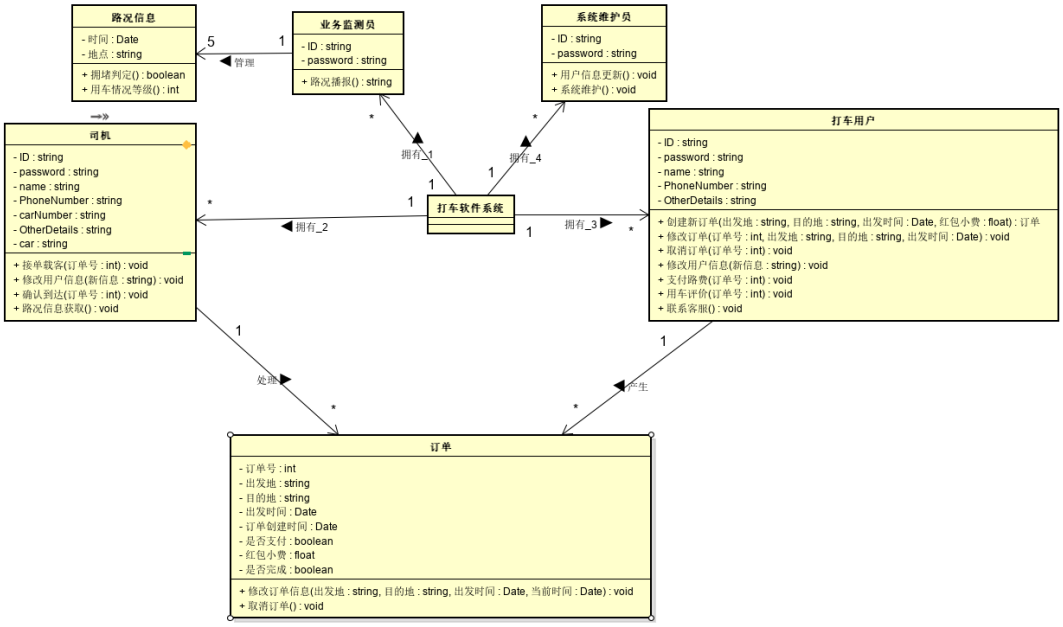
7.1.10.3 过程

业务监测人员可以在系统首页看到订单的分布情况，时间、地点等内容。

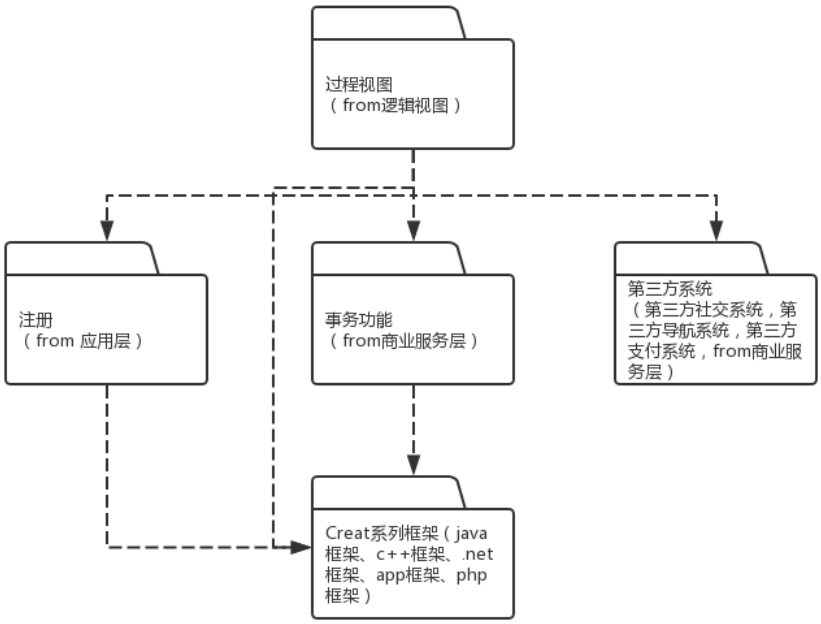
7.1.10.4 输出

显示业务监测人员关注的信息。

7.2 设计元素的过程



7.3 流程模型到设计模型依赖关系

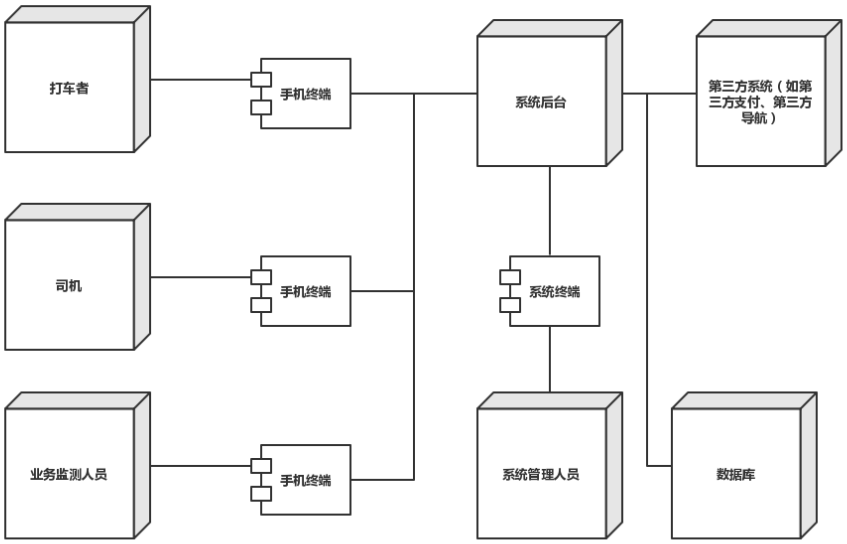


图表 17 依赖关系图

8.部署视图

体系结构的部署视图描述最典型的平台配置的各种物理节点。还描述了任务（从进程视图）到物理节点的分配。此部分按物理网络配置组织;每个此类配置都通过部署图进行说明，然后映射到每个处理器的进程。

本文档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，本系统设计部署视角如下图：



图表 18 部署视图

8.1 手机终端

打车者、司机、业务监测人员通过使用手机终端访问系统后台内容。

8.2 系统后台

各类用户通过不同的终端访问系统后台的不同内容。系统后台包括功能实现、数据库连接及第三方链接。

8.3 各类用户

打车者、司机、业务监测人员、系统管理人员等通过不同的终端访问系统后台内容，进而访问数据库中不同内容。

8.4 数据库

数据库内存储有各类用户信息及系统相应数据，各类用户可通过不同终端访问系统后台进而访问数据库中对应内容。数据库设有各种权限，用户类别不同，权限也不同，可访问数据库中内容也不同。

8.5 第三方系统

第三方系统包括第三方支付系统及第三方导航系统，根据不同功能进行不同系统的访问。

9.结构化视图

9.1 概念级体系结构

表示层:

即在最终用户面前的界面。在这一层为用户呈现出各自不同地界面，用户在该层实现各自的功能操作，并且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用户屏幕上，供用户浏览。

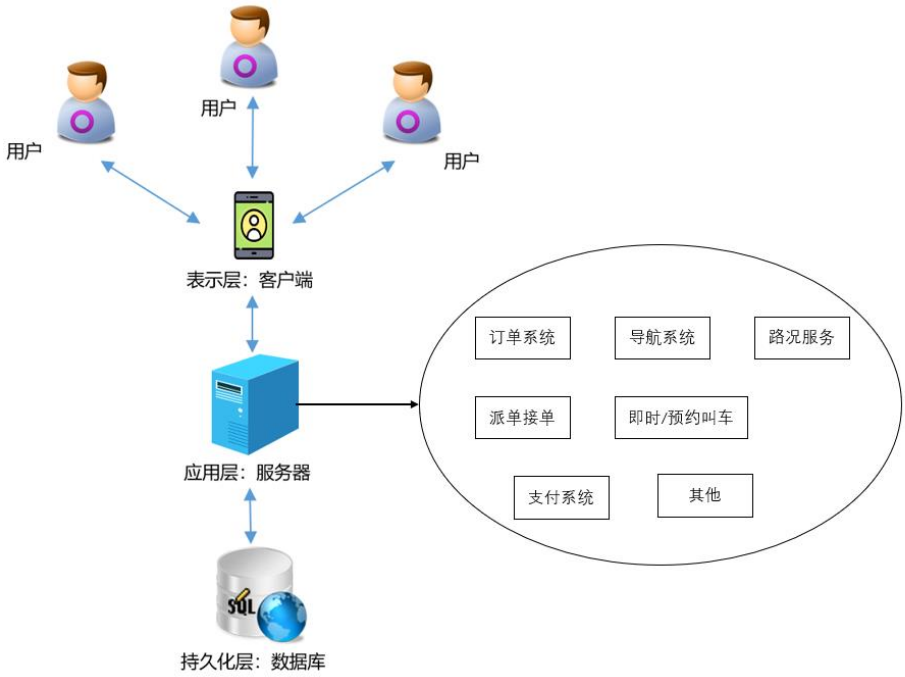
应用层:

即为系统的业务逻辑层，也是系统 WEB 应用服务器的所在位置。在该层接受各种各样的用户请求，并进行处理，然后将结果返回给用户。

数据层:

该层主要负责提供数据信息以及存储数据的功能，能够对不同数据格式以及信息实现共享和交换，返回数据库中的数据或者是修改请求，对数据库中数据进行修改。

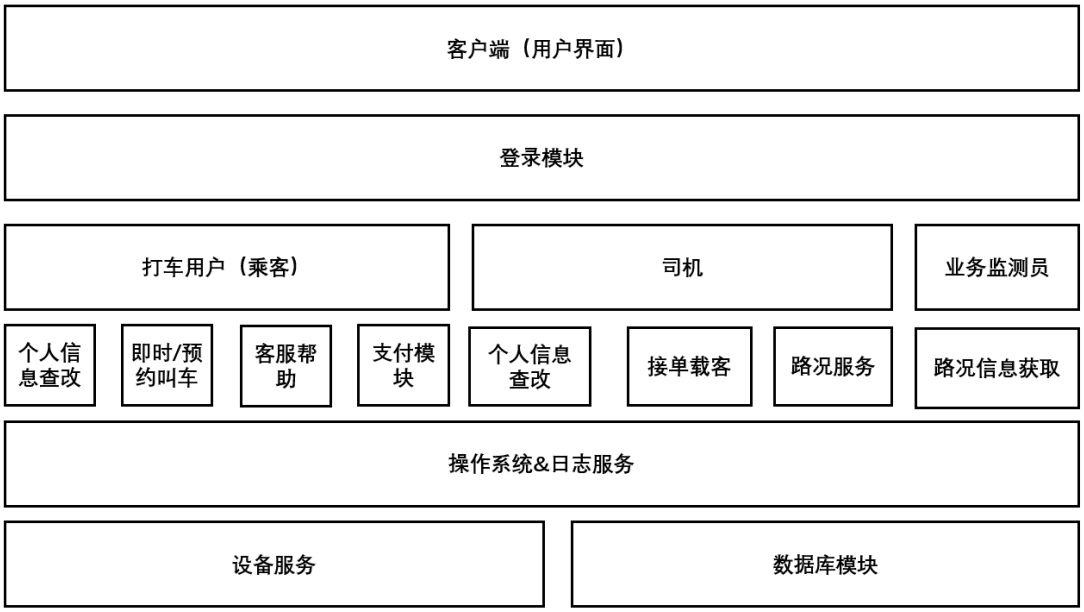
打车软件系统的概念级体系结构图如下所示：



图表 19 打车软件系统概念级体系结构图

9.2 模块级体系结构

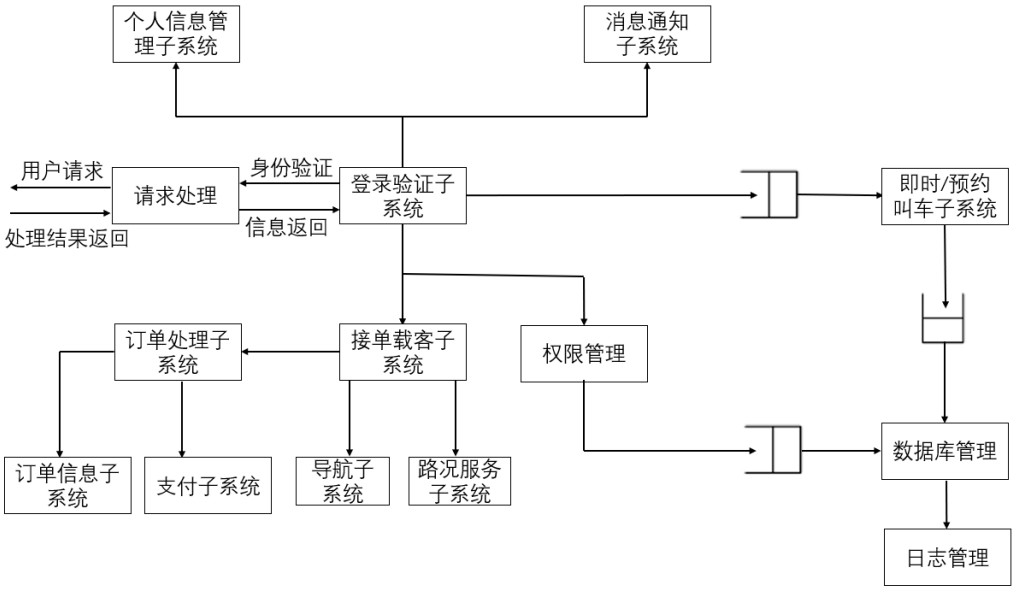
系统的模块化设计是系统进行复用的关键。模块级体系结构反应了对软件代码实现时的期望。特别是对于程序规模较大的系统。体系结构的层次表达了每层的向上一个层面提供的功能和接口，以及需要使用下一层的功能。本说明书根据《打车软件系统需求规格说明书》将系统按功能从逻辑上分解，细化功能结构模块，并按照系统层次进行划分的系统模块化表示，如图所示：



图表 20 打车软件系统模块级体系结构图

9.3 运行级体系结构

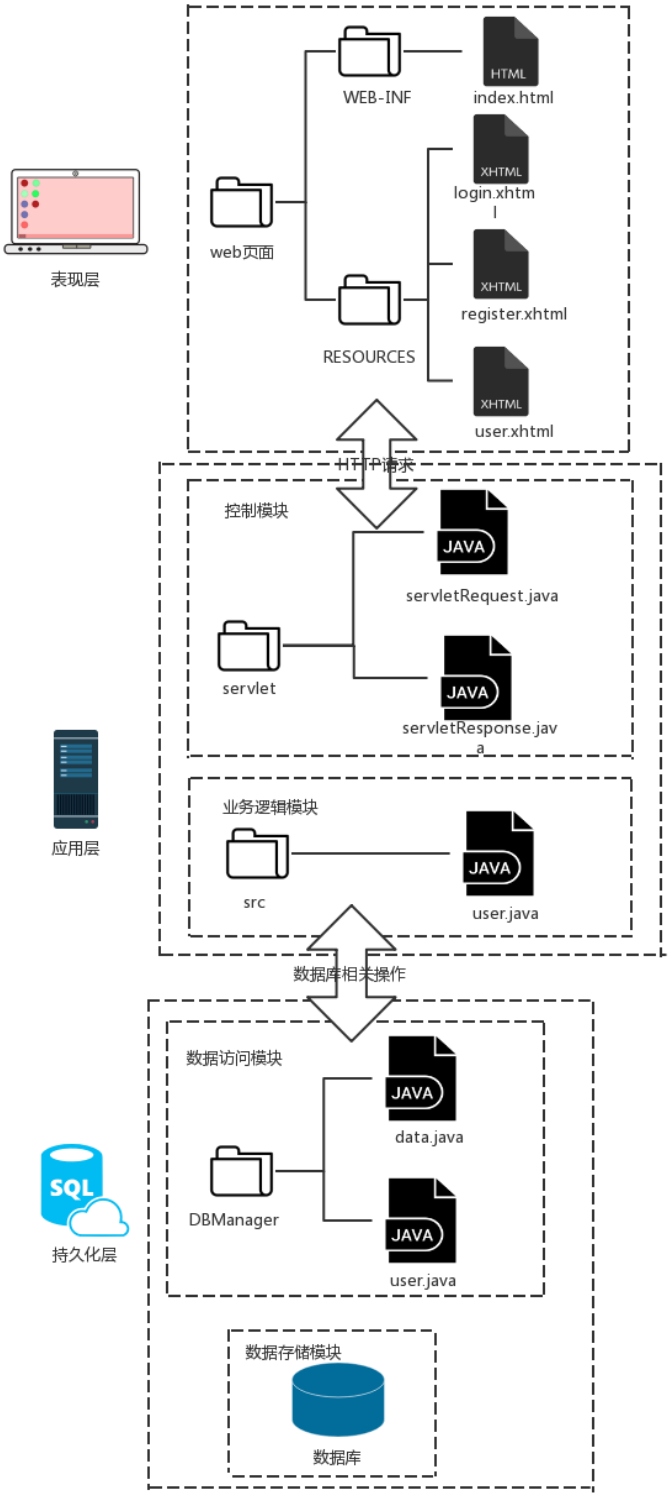
系统的运行级体系结视图描述了模块在运行时，被指派到的特定运行映像运行级体系结构是系统的动态结构，主要用于系统的性能和可调度性分析，以及系统静态与动态配置的管理。系统的消息序列时序图如下：



图表 21 打车软件系统运行级体系结构图

9.4 代码级体系结构

代码体系结构为编程实现提供方便，在编程语言层面按模块、目录、文件、和库的形式组织源代码。如果说在模块化体系结构设计时不要过多地考虑所采用的编程语言的话，这个阶段必须考虑编程语言、开发工具和环境(例如，配置管理)，以及项目和企业的组织结构。从体系结构设计角度看，如何降低代码之间、子项目之间的相互依赖性，提升代码的可重用性等，成为体系结构设计阶段必须考虑的问题。本打车软件系统的代码级体系结构如下图所示：



图表 22 打车软件系统代码级体系结构图

10.质量分析和评价

10.1 质量要求

10.1.1 尺寸与性能要求

本系统主要面对打车用户（乘客）、司机，解决当前打车难、出租车体验差等情况，使得用户乘车体验更加良好，出行更加方便，收费内容一目了然。并且方便司机操作，功能齐全，使用性比较高。同时，因为所存储的订单信息属于比较重要的信息，所以还是要求系统的稳定性和安全性达到一定的高度，出错率降低到一定的水平。才能保证软件的正常、快速、高效的使用。

所以本系统在满足上诉要求情况下应当能够：

1）打车用户（乘客）即时叫车/预约叫车响应时间：

在同时有 50000 用户即时叫车/预约叫车的情况下，系统的响应时间不得大于 5s；

在同时有 10000 用户即时叫车/预约叫车的情况下，系统的响应时间不得大于 3s；

在同时有 5000 用户即时叫车/预约叫车的情况下，系统的响应时间不得大于 1s；

2）司机接单载客响应时间：

在同时有 20000 司机接单载客的情况下，系统的响应时间不得大于 5s；

在同时有 10000 司机接单载客的情况下，系统的响应时间不得大于 3s；

在同时有 5000 司机接单载客的情况下，系统的响应时间不得大于 1s；

3）对处理数据量的要求：

对数据的处理量在 G 的级别时系统的响应应当能够在 3s 之内；

4）数据库容量能够存储至少 5000000 条订单记录，但考虑到可能的扩张，本系统的数据库容量设置为能够存储 7000000 条订单记录。

5）软件整体大小应小于 500MB

系统有足够大的缓存空间，保障系统运行流畅。

10.1.2 可靠性

硬件可靠性：

硬件的可靠性分为网络设备可靠性和服务器硬件可靠性两部分：

1）网络设备的可靠性由公用网和局域网及其设备组成，其中局域网及其设备的可靠性是整个系统的重要指标之一，在设备选择时应考虑到所选设备的可靠性及其售后服务质量。

2）服务器的可靠性是整个系统最基本的可靠性指标，对服务器的选择应充分考虑其可靠性和售后服务质量，更重要的是服务器应该有冗余配置或备份设备，以保障数据库服务器可每天 24 小时运行,其平均故障恢复时间不应大于每 3 个月 1 小时。

软件可靠性：

软件的设计采用面向对象和模块化设计方法，以提高软件模块的独立性和可靠性，从而达到提高软件可靠性的目的。在设计软件时，认真贯彻软件工程中有利于提高可靠性的原则和方法；对运行中可能发生的故障应能自动排解。同时在进行测试时，应当能保证在 100 万用户的并发下，连续 200 天的工作情况下，系统不会出现停止服务的情况。在 100 万次服务请求的情况下，其出错率不得高于 10 次。因软件导致系统崩溃的恢复时间不超过一年 12 小时。

10.1.3 可用性

因为此系统的使用用户是广大打车用户（乘客）、司机，其并不是专业人员，对于复杂系统操作不是很熟悉，为了让用户能够快速的掌握系统的使用方法，系统应当具有设计简洁的图形化界面，同时对于系统的操作应当标上文字已便于理解。同时系统的界面可能需要不断修改完善，我们要求采用 MVC 标准架构。同时系统应当具有一定的容错性，当用户做出一些非法操作时应当给出警告，并且当用户真的执行时不至于引起系统停机或数据混乱。系统必须 24*364 运行，可使用性达到 99%以上。

10.1.4 安全性

1）系统提供用户登录功能(进行用户身份验证或使用微信号作为用户信息)，并且用户名和用户编号是唯一的。用户在登录界面上填写任意的用户名和用户密码(中文或英文)；系统提供登录过程中的出错处理机制和操作成功处理机制。
2）在用户登录时会根据用户的用户名区分不同的用户类型（只可以查询还是既可以查询也可以预定)，判断不同的用户身份。系统管理员会授予不同类型的用户不同的权限。
3）系统对非法用户具有警告功能，例如：单用户表中不存在的用户企图登录系统，系统应该要求用户输入合法用户名和用户密码，并警告用户的操作。
4）为了保障数据信息的安全性，应考虑防电磁辐射，重要的服务器硬件设备的电磁兼容性应满足国家相关标准的要求。
5）操作系统的安全稳定是整个系统的核心，操作系统应具有防病毒措施。所以操作系统的安全水平应在不影响系统功能的情况下尽可能地考虑信息媒体的安全性。首先内部人员对数据的访问操作要进行控制,对用户、信息及操作进行分类授权,防止越权操作,避免数据遭到破坏。信息系统应具有防病毒措施。
6）建设一个成功的自动备份系统，来承担系统的数据备份，在数据库信息遭受意外丢失后可以及时恢复。

10.1.5 可维护性

由于系统涉及的订单信息比较多，数据库中的数据需定期修改，系统可利用的空间及性能也随之下降，为了使系统更好地运转，可以在所有订单信息有较大调整的时候对系统数据及一些简单的功能进行独立的维护及调整。同时要求系统 1 年的停机维护时间不得超过 12 个小时。

10.1.6 可移植性

本系统应当具有一定的跨平台和环境的能力。为了可以作为市面公司打车软件系统的子系统，应当容易与其在同一平台上运行，而不发生冲突。

1. 使用跨平台 Java 语言进行系统的编写，并使用开源库和开源架构。

2. 系统接口易于调用和改造，可以方便地移植到不同的设备上。

10.2 场景分析

10.2.1 用例场景

1. 系统应该能够支持 30000 名用户的并发操作，该场景代表了用户期望的系统性能与效率

2. 用户登录系统的响应时间不超过 1s，用户查询插入数据的请求响应时间不超过 1s，该场景代表了用户期望的系统性能与效率

3. 当数据库访问异常时或者用户误操作时，均应有相应的错误提示，该场景代表了用户期望的可靠性

4. 当服务器故障发生时，系统能够在 1 分钟内切换到备用服务器上继续工作，该场景代表了用户期望的可靠性

10.2.2 增长性场景

1. 未来可能会出现用户量剧增导致的服务器负载过重的问题，可以考虑使用分布式的服务器集群，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

2. 通过扩充现有数据库规模，新增数据库服务器，降低用户的检索时间

3. 通过对软件系统适配不仅手机包括 ipad 和部分 PC 界面，使得用户可以在各个设备终端上访问本打车软件系统。

场景 1	当用户上升到百万级别时，当前的体系结构无法满足服务场景的需求，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。同时，考虑使用 ECC 进行高峰时间的服务器扩容。
场景 2	当用户访问量超过单一系统承受能力时，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。

场景 3	当数据库查询效率无法达到系统要求时，通过扩充现有数据库表的规模，使用数据库索引功能，提高查询效率。考虑到传统数据库可能无法满足系统要求的数据快速更新在要求，在要求具有高查询效率的应用与数据上使用 NOSQL 数据库，尤其针对流式数据、非结构化数据提高查找效率。
场景 4	为了更好地服务客户，针对业务监测员用户增设网页版本打车软件系统的开发，而不仅仅局限在手机客户端，使得业务监测员能够更好地看到数据信息。

表格 7 增长性场景

10.2.3 探索性场景

探索性场景是推动系统封装和降低工作压力的场景。场景的目标是解释当前设计的边界条件的限制，揭露出可能隐含着的假设条件。探索性场景在某种程度上可以为未来打车软件系统的修改给出更实际的需求。

场景 1	优化算法，改进系统的性能，降低系统的响应时间。
场景 2	改进系统的可使用性，从 98%提升到 99.999%。
场景 3	正常情况下，当一半服务器宕机时，不影响整个系统的可使用性。
场景 4	用户能够在浏览器和手机客户端、平板客户端进行相关的操作。
场景 5	提到系统对高并发访问的处理能力。

表格 8 探索性场景

10.3 原型分析

10.3.1 原型法

使用原型法进行需求分析的流程如下：

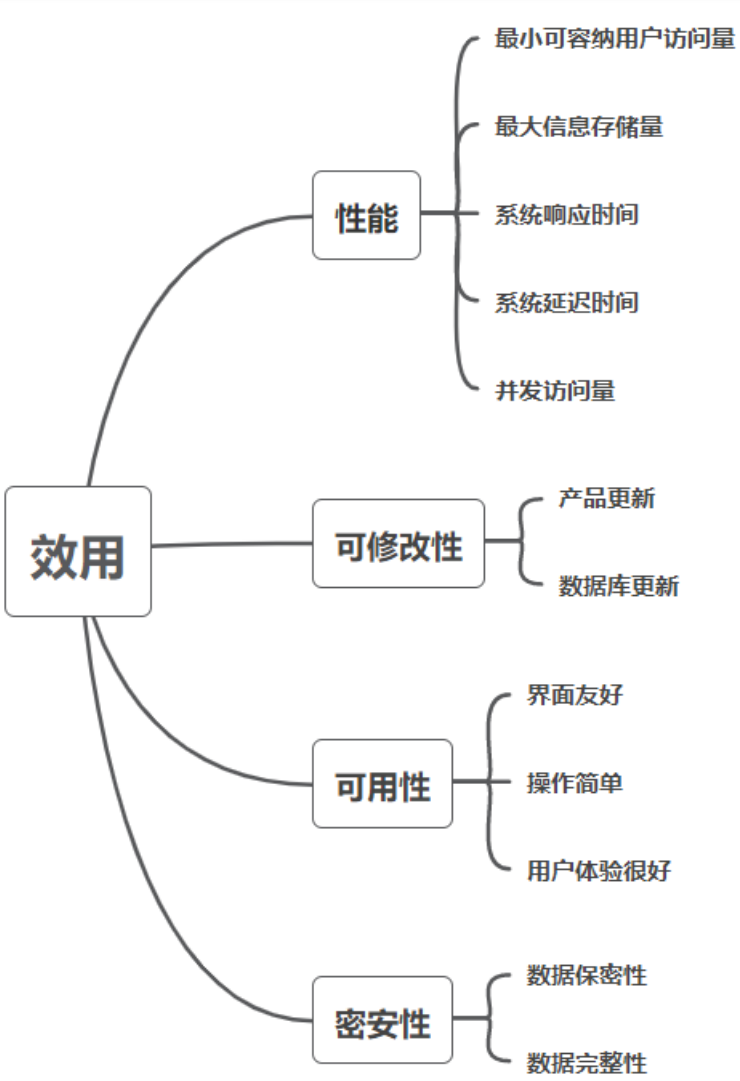
1. 快速分析，列出用户的基本信息需求：打车软件系统的用户可分为打车用户（乘客）、司机、业务监测员三种角色。其中乘客的需求是能够即时预约叫车，用车评价等；司机的需求是查询附近下单信息，接单载客，查看订单、消息，获取路况信息等。业务监测员的需求是实时获得路况信息，并进行处理告知司机、乘客。
2. 构造原型，开发初始原型系统：原型系统先考虑原型系统应必备的特性，暂时不考虑系统的安全性、健壮性，建立基本满足打车用户（乘客）、司机、业务监测员使用要求的系统；

用户和开发人员共同评价原型：本步骤要求系统的使用者（打车用户（乘客）、司机、业务监测员）使用原型进行初步的评价，还要求开发人员对原型系统进行最基本的测试工作，从而找出系统需要改进的地方并进一步进行完善。

10.3.2 效用树

效用树（utility trees）提供自顶向下的机制，直接和有效地将系统的业务具体分解到质量的属性场景，效用树的输出是场景的优先权列表，可供评估人员在短时间内发现体系结构中存在的问题。

本打车软件系统的质量效用树如下：



图表 23 效用树图

10.4 风险

在开发新的软件系统的过程中，由于存在许多不确定因素，软件开发失败的风险是客观存在的。在系统运行过程中，由于外部环境变化或者自身设计的不足，也可能产生各种风险。因此，风险分析对于软件项目管理是决定性的。

10.4.1 技术风险

在软件项目开发和建设的过程中，战略管理技术因素是一个非常重要的因素。项目组一定要本着项目的实际要求，选用合适、成熟的技术，千万不要无视项目

的实际情况而选用一些虽然先进但并非项目所必须且自己又不熟悉的技术。如果项目所要求的技术项目成员不具备或掌握不够，则需要重点关注该风险因素。重大的技术风险包括：软件结构体系存在问题，使完成的软件产品未能实现项目预定目标;项目实施过程中才用全新技术，由于技术本身存在缺陷或对技术的在掌握不够深入，造成开发出的产品性能以及质量低劣。

预防这种风险的办法一般是经常和用户交流工作成果、品牌管理采用符合要求的开发流程、认真组织对产出物的检查和评审、计划和组织严格的独立测试等。软件质量的保证体系是软件开发成为可控制过程的基础，也是开发商和用户进行交流的基础和依据。所以制定卓有成效的软件质量监督体系，是任何软件开发组织必不可少的。

10.4.2 进度风险

软件的工期常常是制约软件项目的主要因素。软件项目工期估算是软件项目初期最困难的工作之一。很多情况下，软件用户对软件的需求是出于实际情况的压力，希望项目承担方尽快开发出软件来。在软件招标时，开发方为了尽可能争取到项目，对项目的进度承诺出已远远超出实际能做到的项目进度，使项目在开始时就存在严重的时间问题。软件开发组织在工期的压力下，往往放弃文档的编写与更新，结果在软件项目的晚期大量需要通过文档进行协调时，却拖累软件进度越来越慢。此外，由于用户配合问题、资源调配等问题也可能使软件项目不能在预定的时间内完成任务。软件项目过程中有自身的客观规律性，用户对软件项目的进度要求不能与软件开发过程的时间需要相矛盾。

对于这种风险解决方案一般是分阶段交付产品、增加项目监控的频度和力度、多运用可行的办法保证工作质量避免返工。在项目实施的时间进度管理上，需要充分考虑各种潜在因素，适当留有余地;任务分解要详细，便于考核;在执行过程中，应该强调项目按照进度执行的重要项，再考虑任何问题时，都要经保持进度作为先决条件;同时，合理利用赶工期及快速跟进等方法，充分利用资源。应该避免：某方面的人员没有到位，或者在多个项目的情况下某方面的人员中途被抽到其他项目，或身兼多个项目，或在别的项目中无法抽身投入本项目。为系统测试安排足够的时间，能使项目进度在改变之初就被发现，这对及时调整项目进度

至关重要。在计划制定时就要确定项目总进度目标与分进度目标;在项目进展的全过程中，进行计划进度与实际进度的比较，及时发现偏离，及时采取措施纠正或者预防，协调项目参与人员之间的进度关系。

10.4.3 质量风险

任何软件项目实施过程中缺乏质量标准，或者忽略软件质量监督环节都将对软件的开发构成巨大的风险。有些项目，用户对软件质量有很高的要求，如果项目组成员同类型项目的开发经验不足，则需要密切关注项目的质量风险。矫正质量低下的不可接受的产品,需要比预期更多的测试、设计和实现工作。

预防这种风险的办法一般是经常和用户交流工作成果、品牌管理采用符合要求的开发流程、认真组织对产出物的检查和评审、计划和组织严格的独立测试等。软件质量的保证体系是软件开发成为可控制过程的基础，也是开发商和用户进行交流的基础和依据。所以制定卓有成效的软件质量监督体系，是任何软件开发组织必不可少的。

A. 附录

A.1 附录一：表对应页码

A.1 附录二：图对应页码