打车软件

体系结构设计文档

4.0

2019/12/7

王淳铮

2017211958

体系结构工程师

# 

**修订历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **描述** | **作者** | **说明** |
| 2019.12.7 | 版本1.0 | 王淳铮 | 模板研读，初始框架搭建 |
| 2019.12.9 | 版本2.0 | 王淳铮 | 了解“4+1”视图建模方法，建立用例视图和逻辑视图 |
| 2019.12.10 | 版本3.0 | 王淳铮 | 实现视图和进程视图的绘制 |
| 2019.12.12 | 版本4.0 | 王淳铮 | 剩余部分的编写，文档的整体完善 |

**审批历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **签名** | **姓名** | **名称** | **日期** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 介绍 5](#_Toc27030)

[1.1 目的 5](#_Toc27853)

[1.2 范围 5](#_Toc30423)

[1.3 定义、术语及缩略 6](#_Toc7258)

[1.4 参考文献 6](#_Toc9439)

[2. 体系结构表示 6](#_Toc5203)

[2.1 体系结构设计步骤 6](#_Toc3681)

[2.2 体系结构视图 8](#_Toc23721)

[3. 体系结构设计目标与约束 8](#_Toc15207)

[3.1关键功能设计目标 8](#_Toc1433)

[3.1.1 乘客功能 9](#_Toc27186)

[3.1.2司机功能 9](#_Toc20362)

[3.1.3 业务监测员功能 10](#_Toc4108)

[3.1.4系统管理员功能 10](#_Toc16149)

[3.1.5功能结构图 11](#_Toc22972)

[3.2 关键质量设计目标 11](#_Toc26164)

[3.2.1性能 11](#_Toc28790)

[3.2.2可靠性 12](#_Toc13279)

[3.2.3易用性 12](#_Toc12165)

[3.2.4密安性 12](#_Toc26021)

[3.2.5可维护性 13](#_Toc6139)

[3.2.6可移植性 13](#_Toc21255)

[3.3 约束 13](#_Toc31530)

[3.3.1实现约束 13](#_Toc18583)

[3.3.2设计约束 14](#_Toc22975)

[3.3.3其他约束 14](#_Toc6444)

[4. 用例视图 14](#_Toc9381)

[4.1概述 14](#_Toc32176)

[4.2关键用例 15](#_Toc13856)

[4.2.1 乘客模块 15](#_Toc24650)

[4.2.2 司机模块 16](#_Toc19659)

[4.2.3 业务监测员模块 17](#_Toc22009)

[4.2.4 系统管理员模块 18](#_Toc24401)

[5. 逻辑视图 18](#_Toc25044)

[5.1概述 18](#_Toc5912)

[5.2系统层次模型 19](#_Toc14225)

[5.2.1 用户界面层 19](#_Toc28996)

[5.2.2 业务相关人员层 19](#_Toc7939)

[5.2.3 业务对象层 20](#_Toc16559)

[6.实现视图 21](#_Toc26787)

[6.1概述 21](#_Toc24782)

[6.2打车软件系统实现视图 21](#_Toc9069)

[7. 进程视图 22](#_Toc10397)

[7.1 概述 22](#_Toc1685)

[7.2进程 23](#_Toc26118)

[7.3 由进程设计元素 23](#_Toc21977)

[7.3.1订单相关模块 24](#_Toc11614)

[7.3.2路况播报和导航模块 25](#_Toc32130)

[7.3.3乘客安全保护模块 25](#_Toc26094)

[7.3.4系统管理员功能块 25](#_Toc19282)

[7.4设计模块间的依赖 27](#_Toc930)

[8.部署视图 27](#_Toc5099)

[8.1 概述 27](#_Toc8021)

[8.2 系统部署视图 28](#_Toc20602)

[8.2.1 用户 28](#_Toc27217)

[8.2.2 手机端 28](#_Toc24233)

[8.2.3 系统服务器 28](#_Toc7931)

[8.2.4 系统后端 28](#_Toc7457)

[8.2.5 系统管理员 29](#_Toc24524)

[8.2.6 第三方系统 29](#_Toc12582)

[9.质量的分析与评价 29](#_Toc16119)

[9.1 场景分析 29](#_Toc26134)

[9.1.1 用例场景分析 29](#_Toc15478)

[9.1.2增长性场景 30](#_Toc24013)

[9.1.3探索性场景 30](#_Toc13654)

[9.2 原型分析 30](#_Toc16701)

[9.2.1原型评价法 30](#_Toc22131)

[9.2.2效用树法 31](#_Toc23231)

[9.3 风险 31](#_Toc30213)

[9.3.1 技术风险 32](#_Toc4066)

[9.3.2 进度风险 32](#_Toc4783)

[9.3.3 质量风险 33](#_Toc7880)

[A附录： 33](#_Toc27633)

[随文所附图 33](#_Toc25421)

# 

# 介绍

## 目的

本文档提供了打车软件系统的全面体系结构概述，使用许多不同的体系结构视图来描述系统的不同方面，其目的是捕获并传达在系统上做出的重要体系结构决策，并使用户了解该软件系统的主要功能以及层次结构，便于用户与技术开发人员的理解与交流，熟练最终用户对系统的使用。本文档涉及到的体系结构视图包括：用例视图、逻辑视图、进程视图、实现视图、部署视图。

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。

本文的预期读者为：

● 项目经理：项目经理可以根据该文档了解预期产品的功能，并据此进行系统设计、项目管理。

● 设计人员：对需求进行分析，并设计出系统，包括数据库的设计。

● 开发人员：了解系统功能，据此进行项目开发。

● 测试人员：根据本文档编写测试用例，并对软件产品进行功能性测试和非功能性测试。

● 客户方：了解预期产品的功能和性能，并与分析人员一起对整个需求进行讨论和协商。

## 范围

本文档所分析的软件系统范围如下：

①该软件系统的名字为：打车软件系统(Taxi Hailing Application System，简称THAS）系统适合以下情况：

● 本打车软件的性能最低满足10万名乘客、3万名司机使用。

● 系统应保持乘客和司机基本信息完整，包括乘客的登录手机号、用户名，司机的驾驶车型等；乘客的下单信息、司机的接单信息和业务监测员的派单信息一致无冗余。

● 系统保证密安性，不会泄露用户的信息，也不允许不具有特定权限的用户更改信息。

● 系统推荐使用的DBMS:MySQL。

● 其他大型的数据库系统也可考虑，但费用较高。如：Oracle、DB2。

● 本文档应与系统需求文档、软件项目需求文档等其他文档保持一致。

②该系统使用时的注意事项：

● 该系统属于在 Android/IOS平台下的软件系统，需要在连接网络的条件下才可使用。

● 该系统只供专业的打车出行规划管理公司使用。系统应用的目的是使得乘客在安全出行的基础上，打车过程更为方便快捷；使得司机与从前沿路接单的传统方式相比，提高接单效率，也可以更加方便地查看收入。同时，也方便业务监测人员对订单进行合理分配。

## 定义、术语及缩略

①需求：用户解决问题或达到目标所需的条件或功能；系统或系统部件要满足合同、标准，规范或其它正式规定文档所需具有的条件或权能。

②需求分析：包括提炼，分析和仔细审查已收集到的需求，以确保所有的风险承担者都明其含义并找出其中的错误，遗憾或其它不足的地方。

③“4+1”视图模型：指从四个角度（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，分析其影响系统的上下文和商业目标情况，由此形成了逻辑视图、进程视图、实现视图和部署视图外加用例视图。

## 参考文献

1. 王安生.《软件工程化》[M]. 北京：清华大学出版社，2014.

# 体系结构表示

## 2.1 体系结构设计步骤

软件体系结构设计师不仅仅是给出软件设计的结果，更重要的是按照规范化的设计步骤进行设计，步骤如下：

①分析体系结构需求。

体系结构设计者要与需求分析团队一起，从应用场景中启发功能需求，体系结构设计师需要理解整个系统需求，并保证系统的质量和可信赖性属性被明确和理解，提出系统体系结构的具体要求。

②分解和逐步细化体系结构

根据推动软件体系结构的源动力，选择已有的成熟的体系结构或者框架，并表示出所要求的的子模块，从上面提到的“4+1视图”出发，实例化这些模块，并分配其相应的功能，最终分解出每个具体的可实现模块。

③编写初步的体系结构文档

及时将模块之间的接口以及该模块类型的约束信息写入各个模块的接口文档中，与整个体系结构框架或模式一起形成初步的体系结构文档。

④对系统的体系结构进行评价

依据初步的体系结构文档，组织评价小组对系统的设计进行评价。设计过程就是对上述过程的多次迭代，直至体系结构设计能够满足质量和其他要求，或者对未来的风险都进行了充分的考虑，最终得到了正是的体系结构文档。

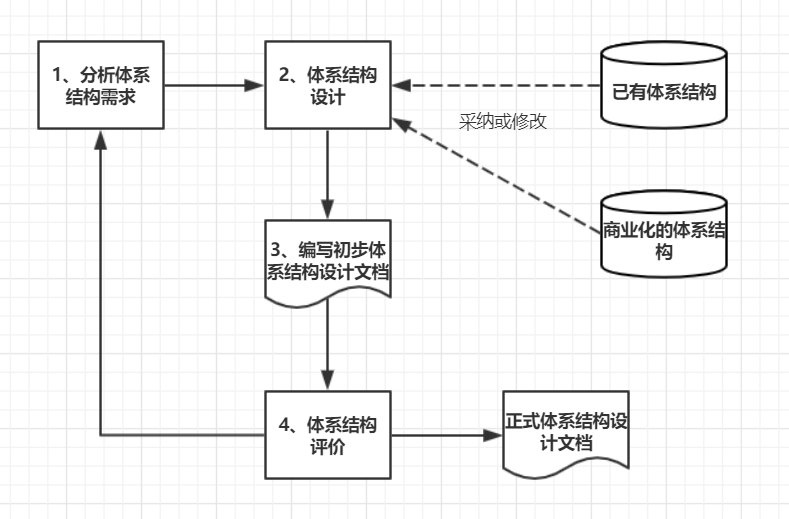


图 2-1 软件体系结构设计步骤

## 2.2 体系结构视图

本文档从4个角度（逻辑、进程、实现和部署）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，分析其影响系统的上下文和商业目标情况，从而形成“4+1”角度看待和理解软件的体系结构，即：逻辑视图、进程视图、实现视图和部署视图，外加用例视图。另外，本文档使用Astah工具进行统一建模语言（UML）模型视图的绘制。

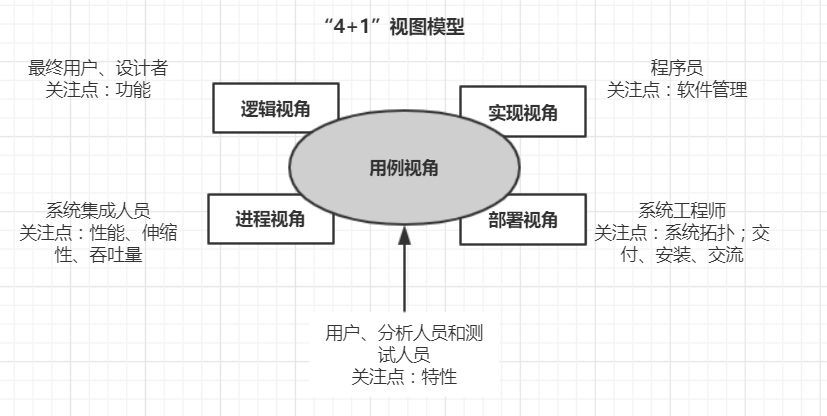


图 2-2 “4+1”视图模型

# 体系结构设计目标与约束

## 3.1关键功能设计目标

打车软件系统可以即时地获取乘客的打车信息，并通过合理的方式将订单分配给司机，相较于传统的沿路挥手叫车的方式，提高了下单和接单的处理效率，为乘客和司机都提供了便利。

经过对系统需求的研究分析，针对不同类型用户，本系统分为四个大的功能块：乘客功能、司机功能、业务监测员功能、系统管理员功能。下面，对该系统的四个功能块的关键功能需求做出定义：

### 3.1.1 乘客功能

可以登录（以乘客的身份）、选择车辆的类型、选择叫车时间、选择上车及下车的地点、选择不同的付款方式、查看当前订单信息、查看历史订单、在指定时间内取消订单；在安全保护方面，可以实名认证，并设置紧急联系人，开启行车录音，发出安全警报等。

①对于乘客的登录，可以选择账号、密码登录，也可以选择绑定的手机号、验证码登录；若用户尚未注册账号，则先注册再登录。

②乘客在叫车时，可以选择出租车、快车或者较为高级的商务用车等不同类型、不同等级的车辆。

③对于叫车时间，乘客可以选择立即乘车，也可以选择三天之内的指定时间乘车。

④乘客可以查找并输入行程的目的地，且默认上车地点为当前所处的位置（考虑到乘客有可能为他人叫车，上车地点可能不是用户所在的当前位置，因此也可以查找指定的上车地点）。

⑤支付方式上，乘客可以选择使用软件内的账户直接支付，也可以通过微信、支付宝等第三方支付方式支付。

⑥乘客可以查看当前订单的地理位置、行驶里程、预估到达时间等信息。

⑦乘客可以查看不同时间段内（例如：一周、一个月、一年内）的历史订单以及账单的汇总信息。

⑧乘客在三分钟之内可以免费取消订单，若超过三分钟，则需承担相应的手续费。

⑨乘客可以添加一个或多个紧急联系人，在发生危险时，迅速发出信号或求救。

⑩乘客可以开启行车录音，以记录车内的情况，更好地保护自身安全。

⑪乘客可以发出安全警报，该警报由业务监测员接收并处理。

### 3.1.2司机功能

可以申请司机资质认证、登录（以司机的身份）、接单、抢单、路程导航、 线上收款、查看历史订单以及账户收入信息。

①对于司机的登录，可以选择账号、密码登录，也可以选择绑定的手机号、验证码登录；若尚未注册账号，或者资质认证尚未通过，则暂时无法以司机身份登录。

②司机可以申请资质，在软件中提交身份证、驾驶证等证件信息，交予系统管理员审核，审核通过以后，即可获得司机资质。

③司机获得订单可以有两种方式，一是接单，由业务监测员综合周边的车辆和乘客情况进行派单，相应的司机便可以获得订单；另一种即为抢单，乘客的订单并不交给业务监测员处理，而是直接发送给一定范围内的司机进行抢单，首先抢到的司机获得本次订单）

④司机可以根据软件中提供的路况等信息，选取从出发地到目的地之间的最佳路线进行导航。

⑤司机可以发起线上收款，向乘客索要车费。

⑥司机可以查看不同时间段内（例如：一周、一个月、一年内）的历史订单以及收入的汇总信息。

3.1.3 业务监测员功能

业务监测员可以派单、播报路况信息、接收处理乘客的安全警报、为乘客设置虚拟号码以保护隐私。

①业务监测员可以收到乘客的订单，并根据等待时间、距离等因素，将订单派发给附近的司机。

②业务监测员可以接收到突发路况信息（例如：交通事故），并将该信息播报给司机，用于规划行车路线。

③业务监测员可以接收到乘客发出的安全警报，并对其进行处理。

④业务监测员可以为乘客设置虚拟号码，司机通过该虚拟号码和乘客联系；也可以对司机隐藏乘客的真实个人信息，保护乘客隐私。

### 3.1.4系统管理员功能

系统管理员可以审核司机的资质和乘客的实名认证信息，并可以在后台登录，查看、修改、删除所有非超级用户的个人信息。

①系统管理员可以审核司机的驾驶证、身份证等信息是否符合条件，从而给司机赋予相应的权限。

②系统管理员可以审核乘客的实名认证信息，完成认证后，可以给乘客赋予相应的高级权限。

③对所有非超级用户的信息进行增删改查。

### 3.1.5功能结构图

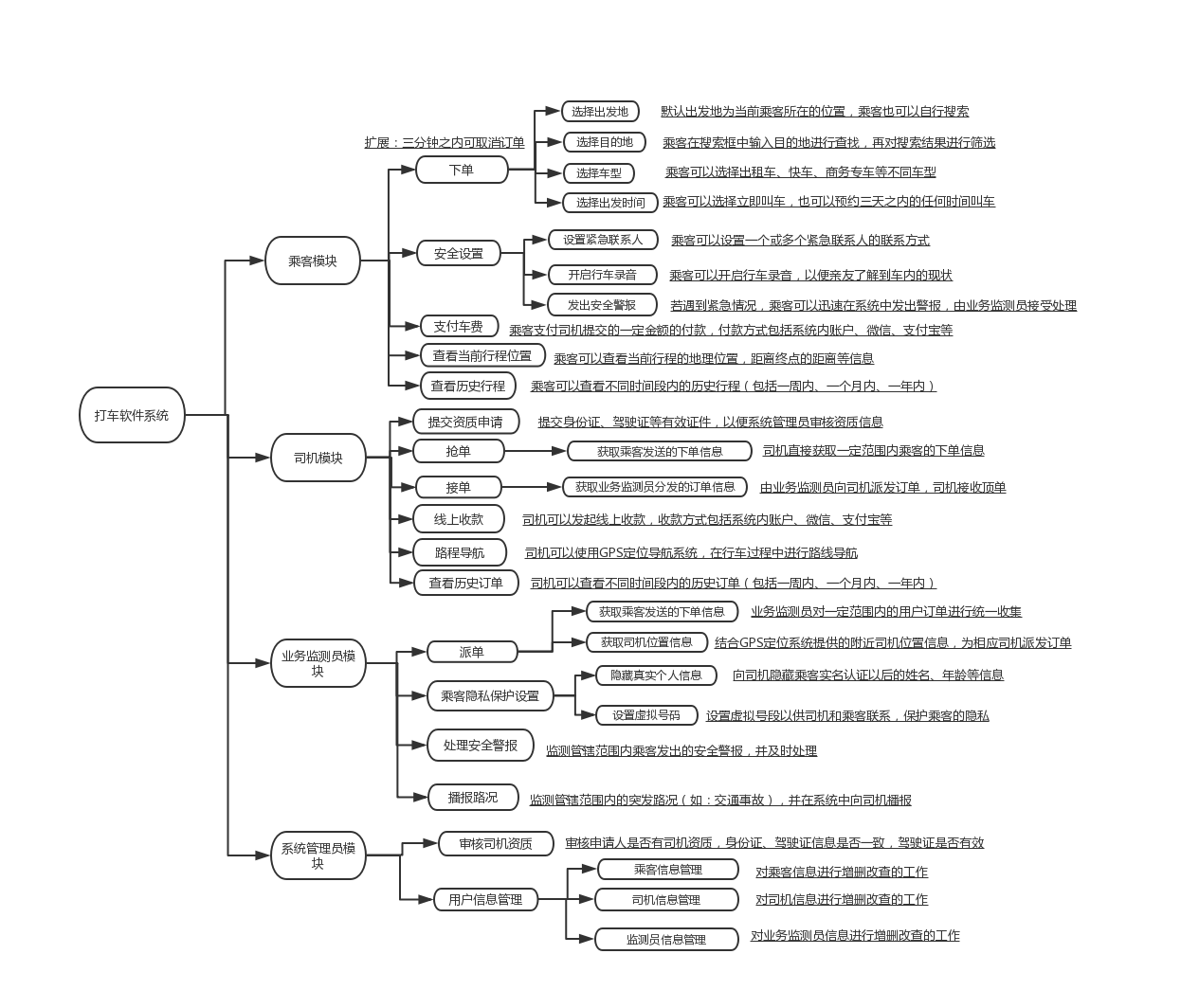


图3-1 功能结构图

## 3.2 关键质量设计目标

质量设计目标包括对性能、可靠性、可靠性、易用性、密安性、可维护性、可恢复性等方面的要求，下面对这六个方面进行详细论述：

### 3.2.1性能

①响应时间：系统在实现查看用户信息、更新行程位置、为司机发布附近乘客订单等功能时，其响应时间最佳为0.5s，平均为1s，不超过3s。

②并发用户数：系统支持并发乘客用户数最少为50000，司机用户数最少为30000。

③资源利用率：在规定条件下，软件产品执行其功能时，使用合适数量和类的资源的能力。

④系统能在高于实际系统运行压力1倍的情况下，稳定地运行12小时以上。

### 3.2.2可靠性

在系统的非功能型需求中，可靠性尤为重要。根据ISO9126的质量定义，可靠性是指在指定使用条件下，软件产品维持规定的性能级别的能力。它又包括成熟性、容错性、易恢复性三个部分。在本项目中，具体体现为：

①除电源硬件、操作系统外不允许异常退出或崩溃。

②系统能较长时间下稳定运行。

③具备一定的故障恢复能力（容错能力），即：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，系统可以进行故障的恢复。

④保证用户在正常使用本系统时，用户的操作或者误操作都不会导致数据丢失。

⑤系统对用户输入的信息有较为严格的验证，保证系统的可靠性。

⑥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。

### 3.2.3易用性

软件的易用性是从用户使用的观点来看，指在特定条件下使用时，软件产品被理解、学习、使用和吸引用户的能力，包括易理解性、易学性、易操作性、吸引性四个部分。

该系统主要面向乘客、司机、业务监测员三大用户，基于不同的用户特点，系统不应忽视用户的满意度，应做到良好的用户交互体验，提供针对不同用户的用户使用说明手册，方便用户使用；另外，还应提供在线帮助界面，方便用户学习操作。

### 3.2.4密安性

①权限控制：根据不同用户角色，设置相应权限，例如：系统管理员可以查看所有非超级用户的信息，乘客、司机只能查看各自的订单信息，不能查看其他用户的订单信息。

②重要数据加密：本系统对一些重要的数据按一定的算法进行加密，如用户口令、重要参数等，也可以通过防火墙保障数据传输的安全。

③数据备份：允许用户进行数据的备份和恢复，以弥补数据的破坏和丢失。

④记录日志：本系统应该能够记录系统运行时所发生的所有错误，包括本机错误和网络错误。这些错误记录便于查找错误的原因。日志同时记录用户的关键性操作信息。

### 3.2.5可维护性

可维护性是指是指软件产品可以被修改和维护的能力。修改可能包括纠错和改进，或者为适应环境、需求和功能规格说明而进行的软件版本升级。包括易分析性、易修改性、稳定性、易测试性四个部分。

在该打车软件中，采用记录日志，用于记录用户的操作及故障信息，同时本系统采用的模式，结构清晰，便于维护人员进行维护。在开发过程中，在软件工程思想指导下，采用规范的开发技术、标准，任用配置管理、质量控制等手段，对文档、计划、编码风格、问题管理等过程与制品进行管理，保证产品的可维护性。

### 3.2.6可移植性

可移植性是指软件产品从一种环境迁移到另一种环境的能力。包括易安装性、共存性、易替换性三部分。

该系统应该可以支持从某一环境转移到另一环境下其相关属性不发生较大改变，功能不发生变化，因此文档定义该系统应选择正确的开发工具，满足对宿主机硬件及操作系统接口发生变化时的适应性，并遵循与可移植相关的标准或约定的软件属性等；系统应该可被容易地卸载，也易被高版本的系统替换。

## 3.3 约束

### 3.3.1实现约束

本系统在进行代码编写的时候要求给出详细的代码注释，每一个功能函数都要给出函数的执行功能，输入和返回值，便于后期进行代码的重用。系统要求应用Java语言进行编写，并且所有变量的命名规范符合Java语言命名规范。系统要求对数据信息进行加密，要求使用当下市面上已有系统的数据加密算法。

### 3.3.2设计约束

系统的最终数据呈现上，一定要求带给用户良好的视觉体验性。约束系统的数据展示清晰、图表美观、相应的功能组件设计得体。这就要求系统最开始的时候从用户的需求出发，设计出相应的应用逻辑可以满足用户的需求，之后根据业务逻辑设计出相应的业务模型开始进行系统的构建，构建的过程之中根据从数据库中获取到的数生成相应视图呈现在网页界面上，呈现给最终用户。

### 3.3.3其他约束

①要求在12个月内完工。

②手机端开发可使用的框架为：XUtil、Retrofit、Butter Knife等。

③服务器系统硬件配置需满足服务器能够高效稳定地与逆行。

服务器需满足：

● CPU:第三代以上双核

● 内存：8G以上

● 硬盘：1T以上

● 可配备磁带机等。

④基于系统安全和保密性的考虑，系统的配置文件、数据存储文件等应进行加密处理，采用国际通用的加密算法，防止意外泄露或恶意攻击。

⑤基于系统可靠性的考虑，系统的数据存储文件应进行冗余备份，比如磁盘冗余阵列存储 RAID 等。

⑥为了系统将来的可扩展性，系统硬件使用国际通用的硬件，不应使用具有针对性的硬件。整个系统也应尽量减少各模块间的调用，尽量做到松耦合。

⑦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5s。

⑧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。

# 用例视图

## 4.1概述

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，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。

## 4.2关键用例

由于本系统主要分为四个功能块：乘客、司机、业务检测员和系统管理员，因此，将关键用例根据不同的用户分为以下四个部分：

### 4.2.1 乘客模块

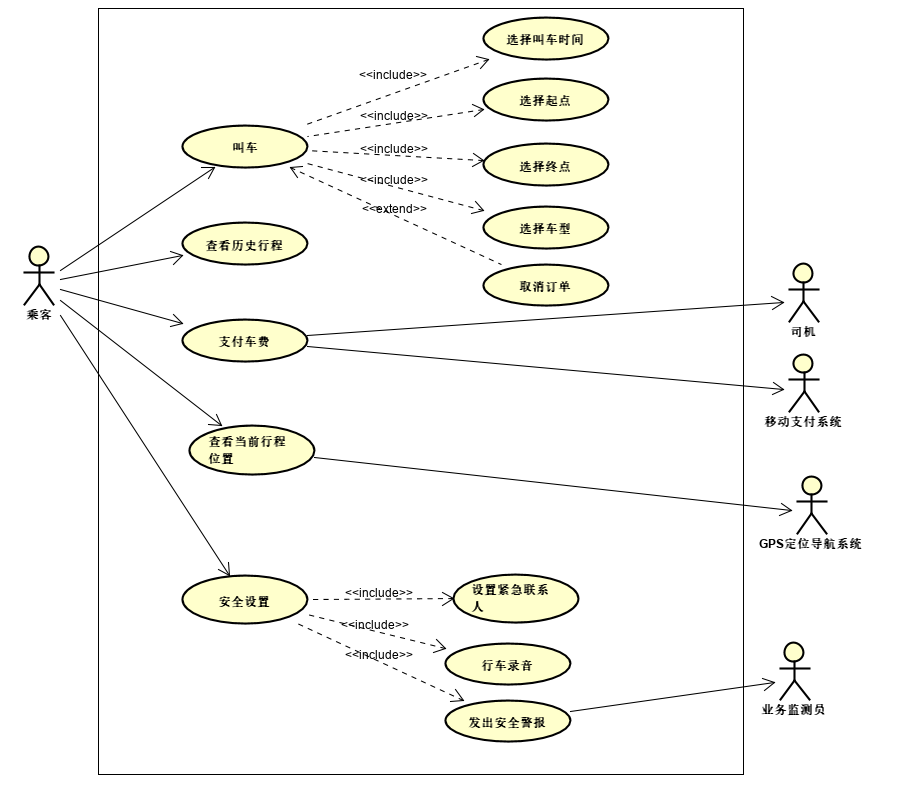


图4-1 乘客模块用例

说明：

|  |  |
| --- | --- |
| 图形 | 含义 |
|  | 参与者 |
|  | 用例 |

图4-2 用例图图例说明

设计论述：

乘客模块包含的子功能有：

* 1. 叫车（选择出发地、目的地、车型、时间）
  2. 查看历史订单
  3. 支付车费
  4. 查看当前订单位置
  5. 安全设置（设置紧急联系人、开启行车录音、发出安全警报）

### 4.2.2 司机模块

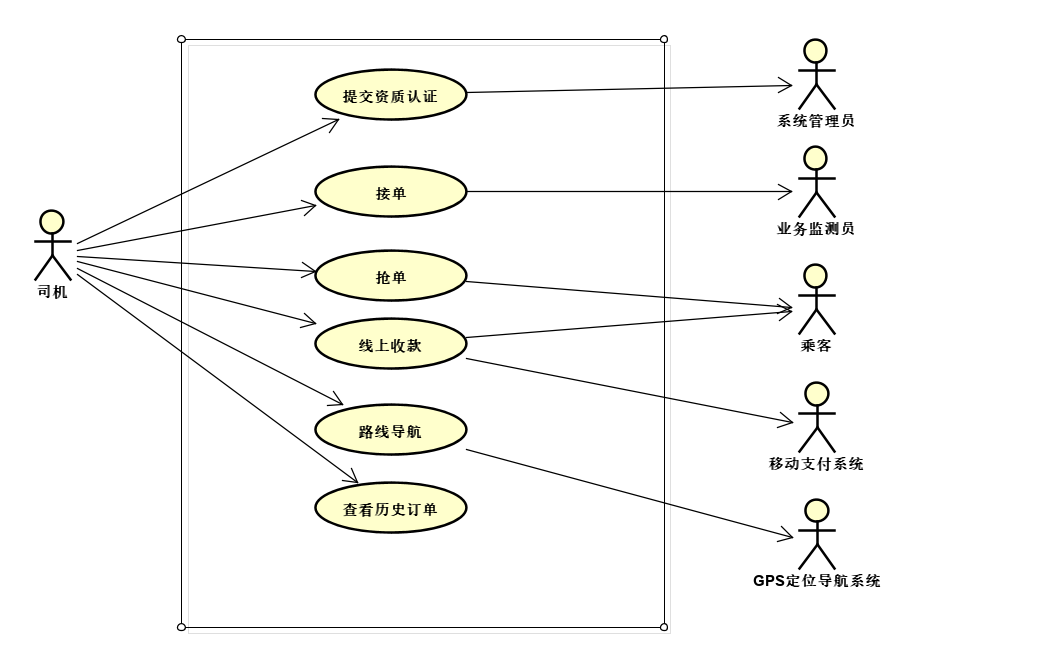


图4-3 司机模块用例

设计论述：

司机模块包含的子功能有：

* 1. 提交资质认证
  2. 接单（由业务监测员派发）
  3. 抢单（直接获取附近用户的订单）
  4. 路线导航（借助GPS导航系统）
  5. 线上收款

### 4.2.3 业务监测员模块

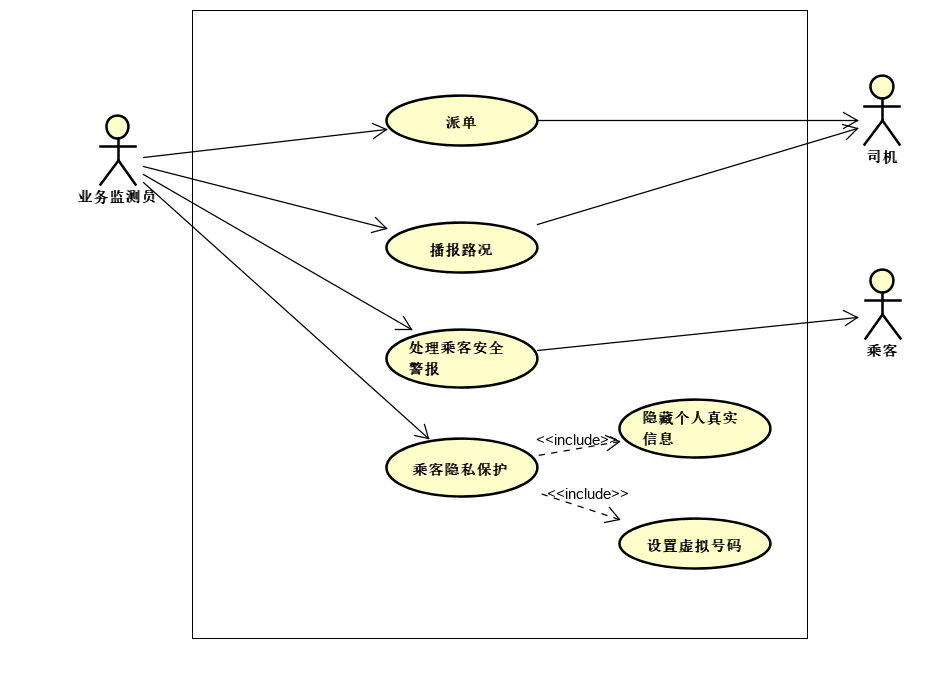


图4-4 业务监测员模块用例

设计论述：

业务监测员模块包含的子功能有：

* 1. 派单（接收一定范围内乘客的订单，将其派发给附近的司机）
  2. 播报路况
  3. 处理乘客安全警报
  4. 乘客隐私保护设置（隐藏实名信息、设置虚拟号码）

### 4.2.4 系统管理员模块

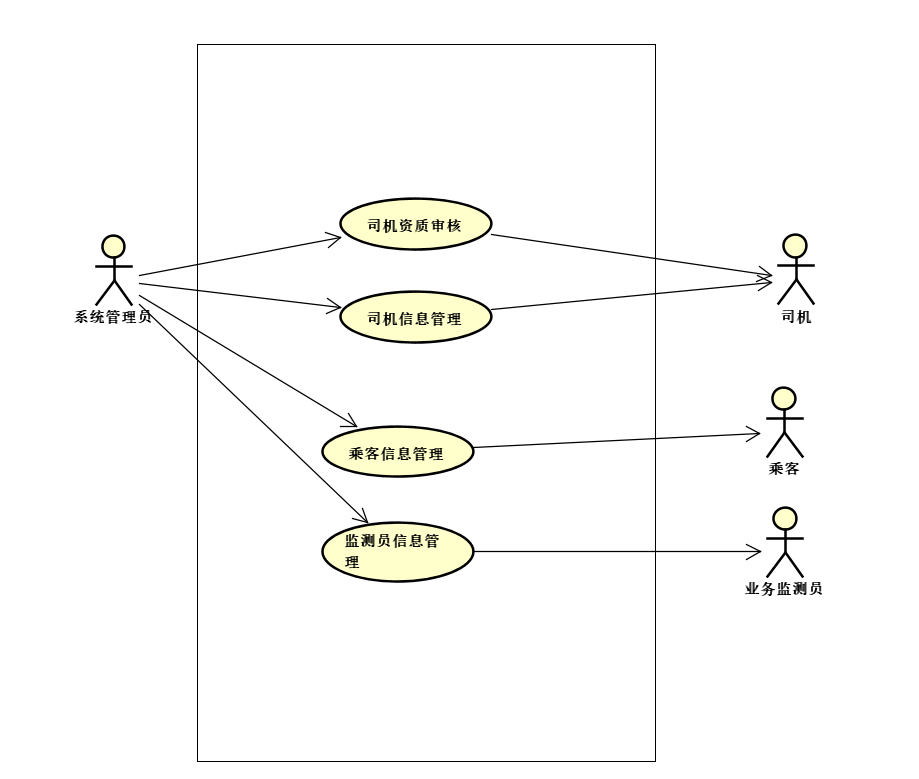


图4-5 系统管理员模块用例

设计论述：

系统管理员模块包含的子功能有：

a)司机资质审核

b)乘客信息管理

c)司机信息管理

d)业务监测员信息管理

# 5. 逻辑视图

## 5.1概述

逻辑视图用来描述系统的功能需求，即在为用户提供服务方面系统所应该提供的功能。在逻辑视图中，系统分解成一系列的功能抽象、功能分解与功能分析，这些主要来自问题领域。在面向对象技术中，表现为对象或对象类的形式，采用抽象、封装和继承的原理。借助于类图和类模板的手段 ，类图用来显示一个类的集合和它们的逻辑关系：关联、使用、组合、继承等。相似的类可以划分成类集合。

对体系结构的逻辑视图的描述，最重要的就是类在服务包和子系统中的组织，以及这些子系统在层中的组织。类图可以用来说明体系结构上重要的类、子系统、包和层之间的关系。

## 5.2系统层次模型

打车软件系统的逻辑视图由三个主要子系统组成：用户界面、业务相关人员和业务对象。

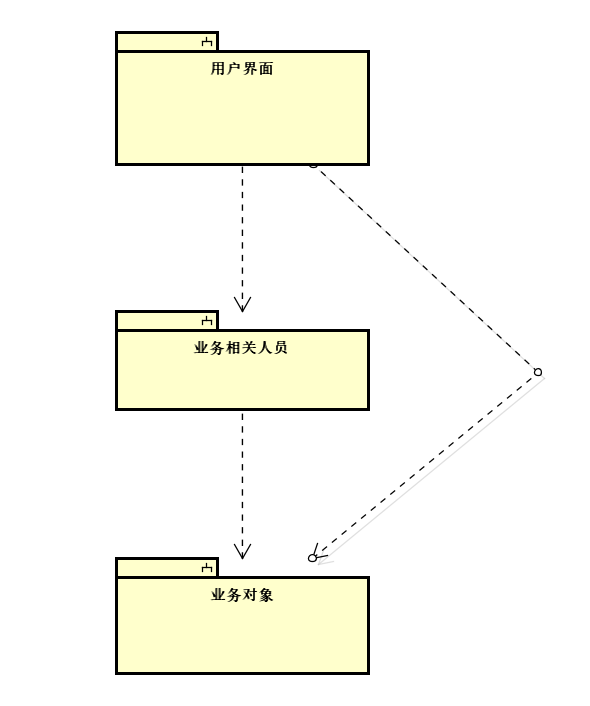


图5-1 子系统的逻辑层次结构

### 5.2.1 用户界面层

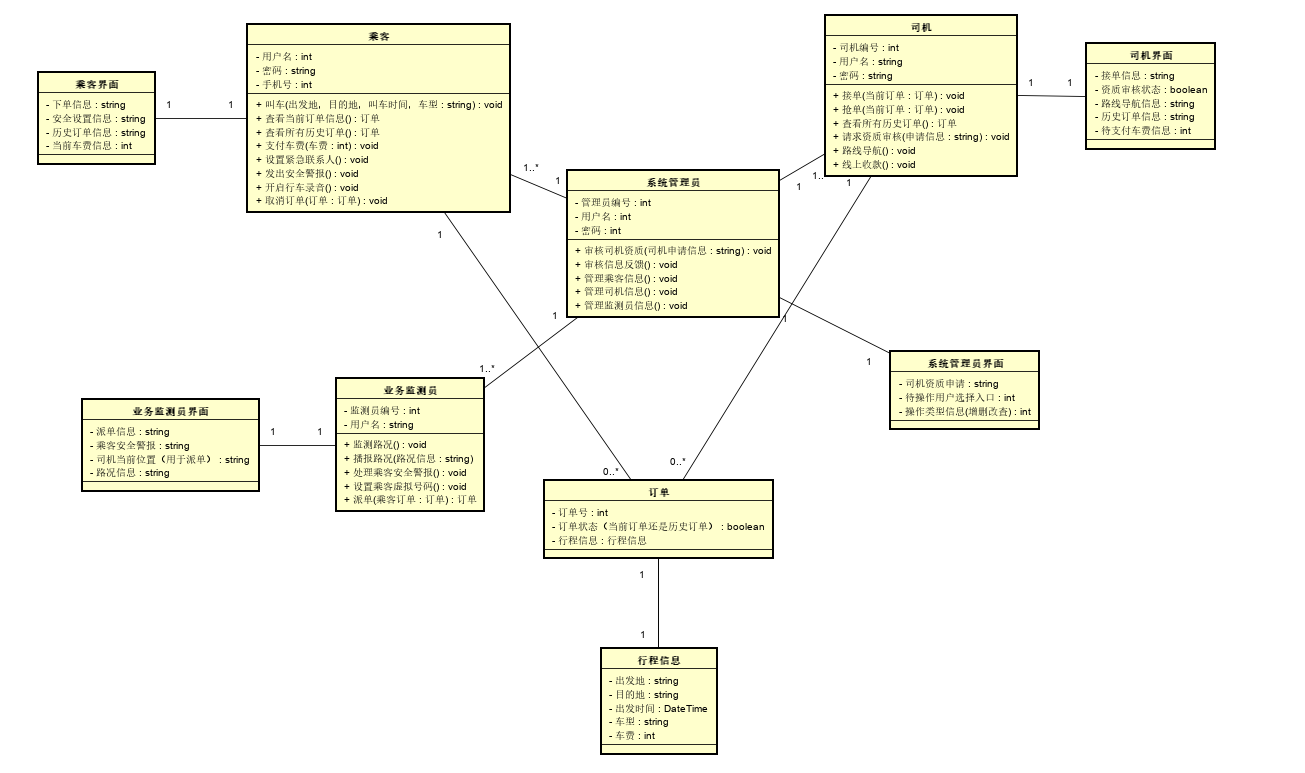
用户界面层即为打车软件的UI组织结构，包括不同类型用户登录的入口，乘客选择叫车信息的提示区域，乘客的安全设置选项，司机接单的入口，业务监测员处理乘客安全警报的入口等。它们是用户与系统之间的接口，是系统的接口边界。

### 5.2.2 业务相关人员层

业务相关人员层包括了本系统的四个主要功能模块：乘客模块、司机模块、业务监测员模块、系统管理员模块，这其中包括了与本系统所使用到的其他系统（例如：移动支付系统、GPS定位导航系统、数据库等）相连接的接口，也包含了整个系统的主要业务逻辑。

### 5.2.3 业务对象层

业务对象层包括系统中涉及到的两个实体类：订单类和行程信息类。司机和乘客在下单和接单时都会生成相应的订单，一个订单中又包含着叫车时间、叫车地点、车型、车费等信息，这些信息又存放在行程信息类当中。

图5-2 系统数据库的逻辑结构

## **6.实现视图**

## 6.1概述

实现视图，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。

系统的开发架构用模块和子系统图来表达，显示了"输出"和"输入"关系。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。开发视图的风格通常是层次结构，每个层为上一层提供良好定义的接口，层次越低，通用性越好。

## 6.2打车软件系统实现视图

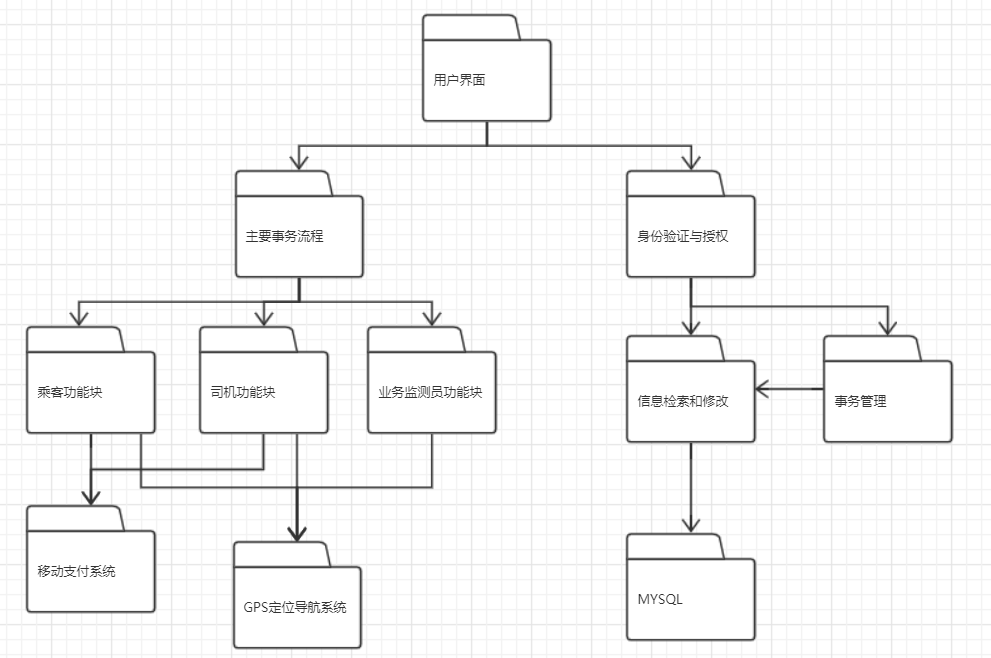


图6-1 打车软件系统实现视图

说明：

|  |  |
| --- | --- |
| 图形 | 含义 |
|  | 包 |

图6-2 实现视图图例说明

# 7. 进程视图

## 7.1 概述

进程架构考虑一些并发性、分布性、系统完整性、容错性的问题，以及逻辑视图的主要抽象如何与进程结构相配合在一起，即定义逻辑视图中的各个类的具体操作是在哪一个线程中被执行。进程视图侧重系统的运行特性，服务于系统集成人员，方便后续性能测试。

进程架构可以在几种层次的抽象上进行描述，每个层次针对不同的问题。在最高的层次上，进程架构可以视为一组独立执行的通信程序的逻辑网络，它们分布在整个一组硬件资源上，这些资源通过 LAN 或者 WAN 连接起来。多个逻辑网络可能同时并存，共享相同的物理资源。

## 7.2进程

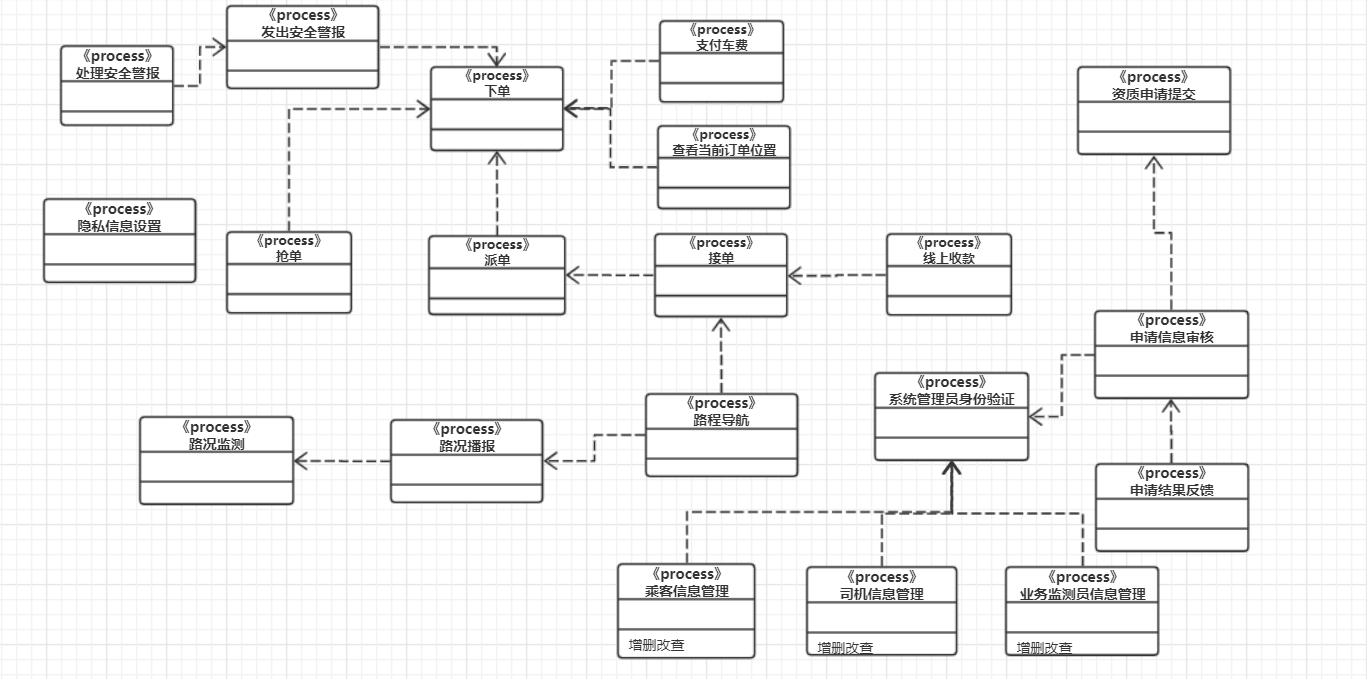


图7-1 进程总览

注：图中的虚线表示进程之间的依赖关系，若进程A需要依赖进程B才能进行，则虚线箭头则由A指向B，箭头含义与普通流程图中的略有不同；对于进程之间依赖关系的详细叙述将在7.3中呈现。

## 7.3 由进程设计元素

从进程的角度来看，整个打车系统包含以下四个大的部分（子系统）：

①订单处理：包括乘客的下单，业务监测员的派单，司机的接单或抢单，司机线上收款，乘客交费这样一个完整流程。

②乘客安全信息保护：包括乘客发出安全警报，业务监测员对安全警报进行处理，以及业务监测员对乘客隐私信息进行设置（如：设置虚拟号码）。

③路况播报及导航：包括业务监测员对实时路况进行监测，并将监测到的路况播报给司机；以及司机根据路况信息调整当前的路程导航。

④系统管理员功能块：包括对司机的资质信息进行审核，并反馈结果；还包括对所有非超级用户的信息进行管理（增删改查）。

完整的元素设计图如下：

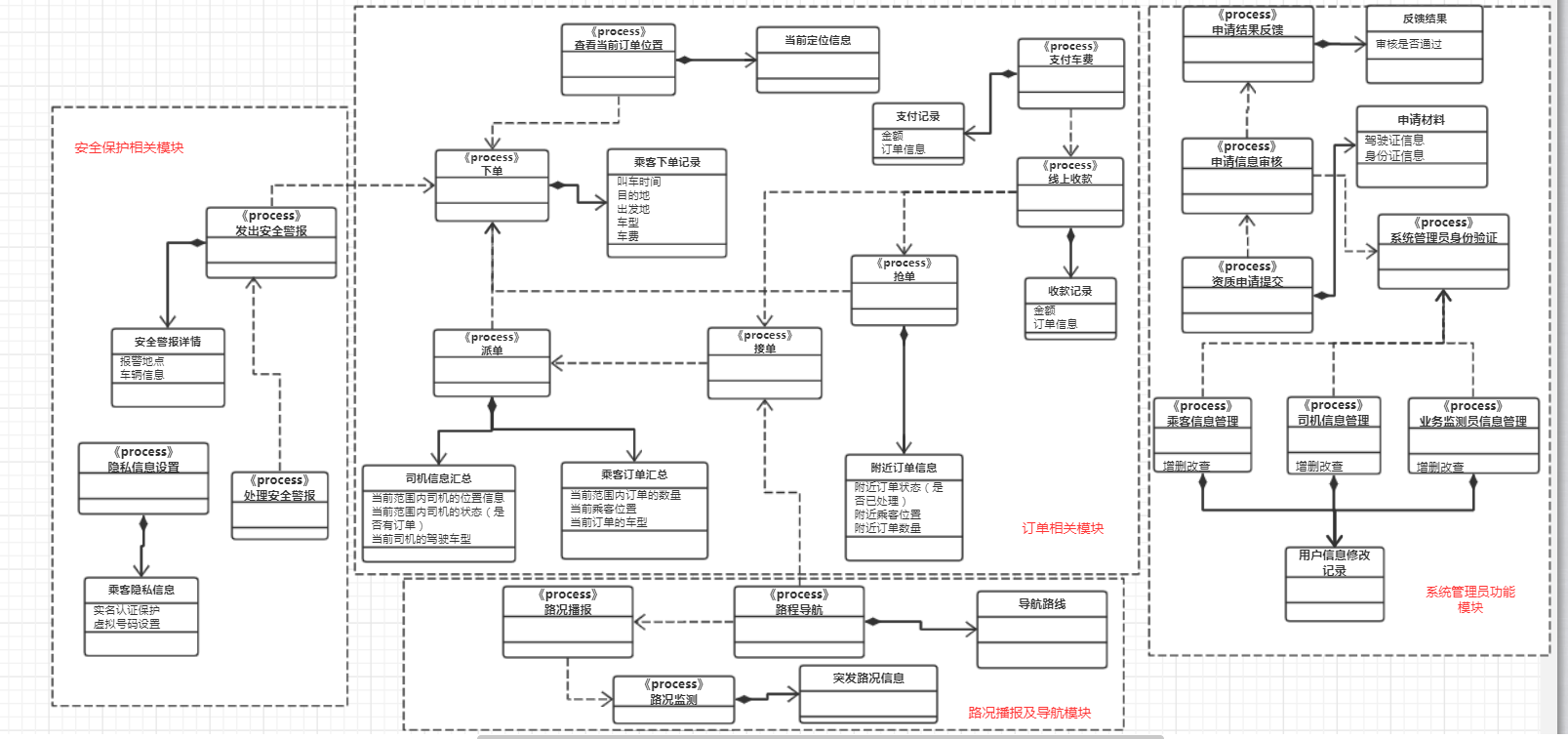


图 7-2由进程设计元素

下面分为这四个模块进行元素的设计：

### 7.3.1订单相关模块

订单相关模块中的进程主要可以分为下单、派单、接单、抢单、查看当前订单信息，支付车费等，这些进程之间也存在着一定的依赖关系等进程，比如说：派单和抢单进程必须依赖于乘客的下单进程（即：乘客下单以后，业务监测员和司机才能进行相应的派单和抢单过程）；接单进程必须依赖于派单进程等，支付车费进程必须依赖于下单进程。

所以完整的过程为：乘客下单进程中会涉及到对于目的地、出发地、叫车时间、车型等信息的选择，因此，下单进程可以分解出乘客下单信息这一元素；业务监测员的派单进程中，业务监测员首先要获取管辖区域内所有乘客的订单以及司机的位置，进而根据司机位置与订单位置的关系进行合理的派单，因此，由派单进程可以分解成乘客订单汇总和司机信息汇总两个元素；在另一种获取订单模式，即：抢单模式中，司机可以自动获取附近乘客的位置及状态信息，因此，由派单进程可以设计出附近订单信息这一元素；最后，乘客在路途中可以实时查看位置，因此，由查看订单位置进程可以设计当前定位信息这一元素；支付车费时会生成一个缴费记录，包括交费金额、乘车出发地、目的地等信息，因此可以设计支付记录这一元素，同样的，司机的线上收款也可以设计收款记录这一元素。

### 7.3.2路况播报和导航模块

该模块中的进程可以分为路况监测、路况播报、路况导航三个部分，其依赖关系为：司机在收到业务监测员的突发路况播报以后，使用GPS导航软件进行路线的及时修改，因此，路线导航进程依赖于路况播报进程；另外，业务监测员只有在监测到路况信息后，才能进行播报，所以路况播报进程又依赖于路况监测进程，此外，路程导航信息也需要依赖于订单模块中的接单进程。该模块中，对于路况导航进程需要设计导航路线这一元素；路况监测进程需要设计突发路况信息这一元素，用来记录具体的突发信息。

### 7.3.3乘客安全保护模块

该模块主要包括发出安全警报、处理安全警报、隐私信息设置进程。在行车途中，若遇到突发情况，乘客可以及时通过软件发出安全警报，业务监测员若发现管辖区域内有警报信息，则立即进行处理，因此，处理安全警报进程需要依赖于发出安全警报进程。在元素的设计上，乘客发出安全警报时，需要设计一个元素来记录警报详情，包括报警地点和车辆信息。在隐私信息设置进程中，需要记录乘客的实名认证保护信息以及设置的虚拟号码，因此要设计乘客隐私信息元素。

### 7.3.4系统管理员功能块

该模块主要包括系统管理员身份认证、资质信息审核、结果反馈以及对不同用户信息进行管理。首先，其他所有进程都必须依赖于身份认证进程，只有身份认证以后才能进行相应的操作。在元素设计上，系统管理员在修改用户信息时要进行相应的记录，因此需要设计用户信息修记录元素；另外，司机在提交资质申请时还要提交驾驶证。身份证等信息，因此要设计申请材料元素；系统管理员的反馈结果进程也要设计反馈结果元素。

说明：

|  |  |
| --- | --- |
| 图形 | 含义 |
|  | 依赖关系（Dependency） |
|  | 组合关系（Composition） |
|  | 设计元素 |
|  | 进程 |

图7-3 进程视图图例说明

● 组合关系：在UML中，组合关系表示为实心菱形箭头线。组合关系强调了比聚合关系更加强的整体/部分的关联，和聚合关系所不同的是，在组合关系中，虽然局部不一定随着整体的销毁而销毁，但整体要么负责保持局部的存活状态，要么负责将其销毁。也就是说，组合关系中，局部的存活期一定是小于，最多是等于整体的存活期的。

● 依赖关系：也是表示类与类之间的连接，表示一个类依赖于另外一个类的定义，依赖关系时是单向的。简单理解就是类A使用到了类B，这种依赖具有偶然性、临时性，是非常弱的关系。但是类B的变化会影响到类A。

在本系统中，依赖关系可以表示为一个进程需要依赖另一个进程才能够执行，例如：业务监测员的派单进程，必须要依赖于乘客的下单进程（即：乘客下单以后，业务监测员才能够派单）。

## 7.4设计模块间的依赖

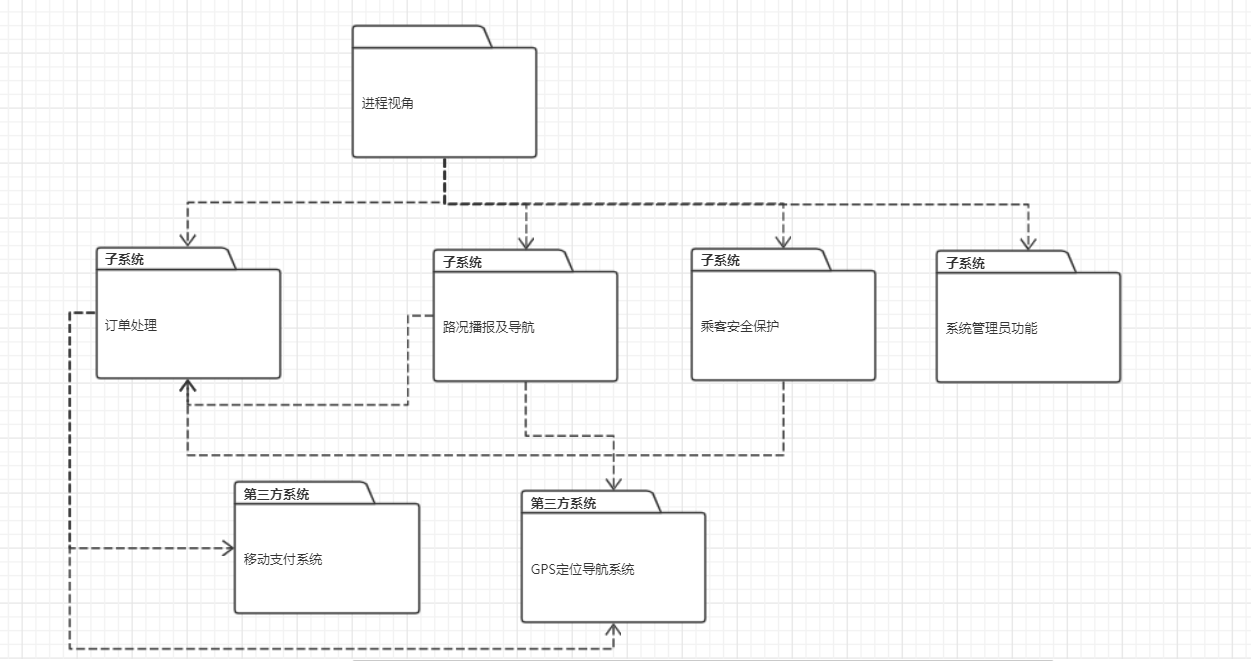


图7-4 设计模块间的依赖

在四个子系统中，路况播报及导航子系统以及乘客安全保护子系统都依赖于订单处理子系统，另外，订单处理子系统还依赖于移动支付的第三方系统以及GPS定位导航第三方系统；路况播报及导航子系统也要依赖于GPS定位导航第三方系统。

注：模板上6.4节给出的由进程到实现的示意图已经在实现视图中单独列出，在此不再重复叙述。

# 8.部署视图

## 8.1 概述

部署视图，也叫物理视图，主要描述硬件配置，服务于系统工程人员，解决系统的拓扑结构、系统安装、通信等问题。主要考虑如何把软件映射到硬件上，也要考虑系统性能、规模、可靠性等。可以与进程视图一起映射。物理架构主要关注系统非功能性的需求，如可用性、可靠性（容错性），性能（吞吐量）和可伸缩性。

软件在计算机网络或处理节点上运行，被识别的各种元素（网络、过程、任务和对象），需要被映射至不同的节点；我们希望使用不同的物理配置：一些用于开发和测试，另外一些则用于不同地点和不同客户的部署。因此软件至节点的映射需要高度的灵活性及对源代码产生最小的影响。

## 8.2 系统部署视图

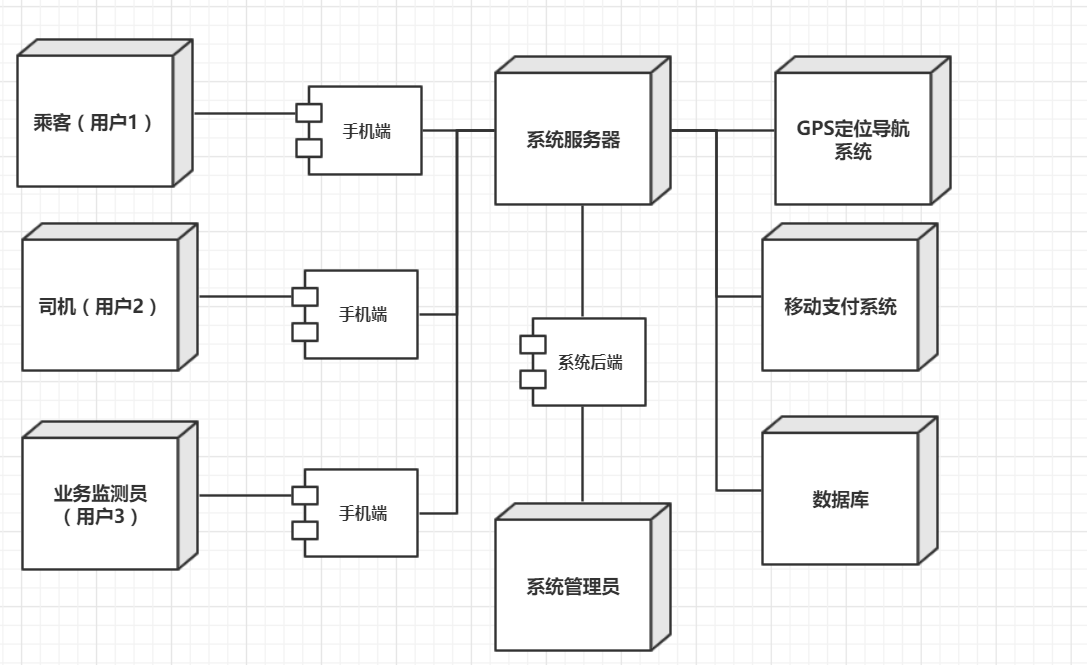


图8-1 系统部署视图

### 8.2.1 用户

打车软件系统的用户包括乘客、司机和业务监测员，它们都可以通过手机端使用系统，满足各自的需求。

### 8.2.2 手机端

由于用户在使用打车软件系统时大都在外出行，不方便使用电脑，因此该系统应该以手机端APP或网页为主要的呈现形式，以满足便捷性的要求。

### 8.2.3 系统服务器

系统服务器是整个系统的核心，包含对主要业务功能的实现。

### 8.2.4 系统后端

系统后端是系统服务器中功能的具体代码实现，通过业务人员的编程来实现具体的业务逻辑。

### 8.2.5 系统管理员

系统管理员负责系统后端的维护，以及对各类非超级用户的信息管理。

### 8.2.6 第三方系统

包括数据库、GPS定位导航系统和移动支付系统等，他们与打车软件系统存在相应的接口，辅助打车软件系统实现其功能。

# 9.质量的分析与评价

## 9.1 场景分析

场景分析通过分析软件应用的场景，从用户的角度出发，从场景的角度来设计测试用例，是一种面向用户的测试用例设计方法。系统的使用场景与质量属性的要求是密切相关的，也是决定体系结构的重要依据。使用场景包括了系统需求工程中所提出的各种需求，通过场景可以很好地评估体系结构。

### 9.1.1 用例场景分析

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：乘客**希望可以快速准确地搜索到自己的出发地和目的地，选择合适的车型，并且可以方便地与司机进行联系。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**希望可以在行车过程中的语音播报清晰明了，并且可以及时接收到突发的路况变化信息。该场景表明了用户期望系统易于使用，即易用性。

**场景3：系统管理员**希望可以在审核司机的资质信息后，系统可以及时清楚地将审核信息反馈给司机。该场景表明了用户期望系统易于使用，即易用性。

**场景4：乘客**在查询目的地位置时，期望可以在3s内获得结果。该场景表明了系统的性能需求。

**场景5**：**业务监测员**希望在乘客发出安全警报后，可以在3s内及时接收。该场景表明了系统的性能需求。

**场景6：**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：**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。

**增长性场景3：**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

**增长性场景4：** 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打车软件系统。

### 9.1.3探索性场景

**探索性场景1：**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2：**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，作为乘客用户，在不断地调用查询条件30000次，查询其目的地或出发地时，看其需要多长时间。

②写一个测试程序，作为系统管理员用户，在不断地调用增删改查等语句3000次，对不同用户的信息进行操作时，看其需要多长时间。

### 9.2.2效用树法

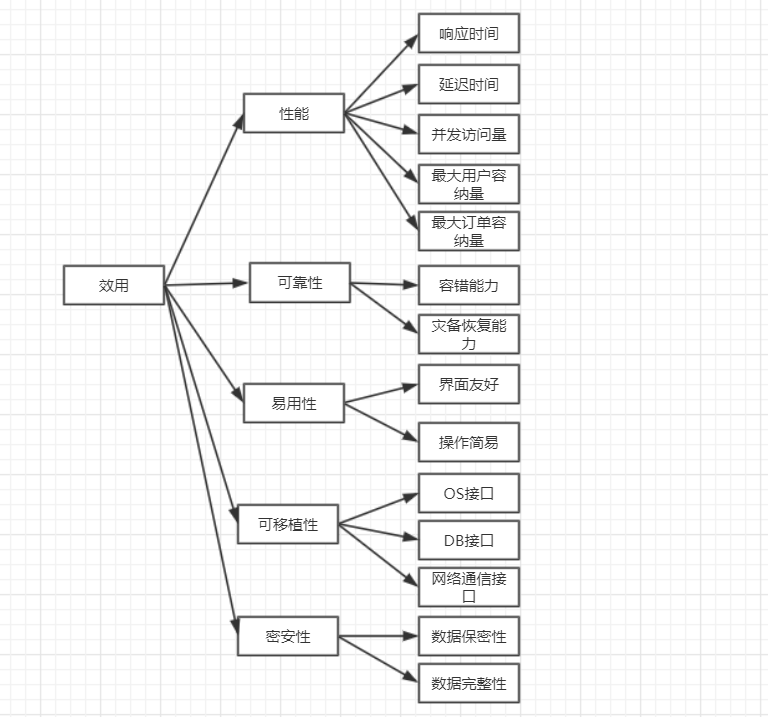


图9-1效用树

## 9.3 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。

### 9.3.1 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。

### 9.3.2 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。

### 9.3.3 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：

随文所附图

图2-1 软件体系结构设计步骤

图2-2 “4+1”视图模型

图3-1 功能结构图

图4-1 乘客模块用例

图4-2 用例图图例说明

图4-3 司机模块用例

图4-4 业务监测员模块用例

图4-5 系统管理员模块用例

图5-1 子系统的逻辑层次结构

图5-2 系统数据库的逻辑结构

图6-1 打车软件系统实现视图

图6-2 实现视图图例说明

图7-1 进程总览

图7-2 由进程设计元素

图7-3 进程视图图例说明

图7-4 设计模块间的依赖

图8-1 系统部署视图

图9-1效用树