[相关信息](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/pt_sad.htm)

**某某打车软件系统**

**软件体系结构文档  
版本 2.0**

**米尔克**

**2017211978**

**修订历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **描述** | **作者** |
| 2019年12月5日 | 1.0 | 由需求分析报告生成初版体系结构文档。 | 米尔克 |
| 2019年12月7日 | 2.0 | 文档进一步完善更新 | 米尔克 |
|  |  |  |  |
|  |  |  |  |

*<<在双尖括号内部的阴影文字不属于此打车软件派车系统需求规格说明书中的正式部分。而是为了帮助读者更好地理解此份需求说明书。>>*

*<<本体系结构说明书是北京邮电大学软件学院2019年秋季学期软件工程导论课实践作业的一部分，其目的是让学生更好地掌握软件工程体系中的需求工程阶段工作以及掌握多种实用的建模方法和图示表达。*

*本体系结构说明书使用授课老师王安生教授提供的模板（王安生教授编写的模板使用须知请见附录1），并在其基础上进行了改编。*

*本期中作业的完成人为米尔克（学号：2017211978 班级：2017211504 邮箱：1317946978@qq.com）>>*

目录

[**1.简介[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 4](#_Toc26708054)

[**1.1目的[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 4](#_Toc26708055)

[**1.2范围[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 4](#_Toc26708056)

[**1.3定义、缩略语和缩写[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 5](#_Toc26708057)

[**1.4参考文献[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 5](#_Toc26708058)

[**2.架构表现[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 6](#_Toc26708059)

[**2.1逻辑视角** 7](#_Toc26708060)

[**2.2 进程视角** 8](#_Toc26708061)

[**2.2.1 打车者进程** 8](#_Toc26708062)

[**2.2.2司机进程** 9](#_Toc26708063)

[**2.2.3 业务监测人员进程** 10](#_Toc26708064)

[**2.2.4 系统管理人员** 10](#_Toc26708065)

[**2.3 实现视角** 11](#_Toc26708066)

[**2.4 部署视角** 11](#_Toc26708067)

[**3.架构目标和约束[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 12](#_Toc26708068)

[**3.1架构目标** 12](#_Toc26708069)

[**3.2 设计约束** 12](#_Toc26708070)

[**3.3 逻辑数据库要求** 12](#_Toc26708071)

[**3.4其他约束** 13](#_Toc26708072)

[**4.用例视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 13](#_Toc26708073)

[**4.1 用例1——打车者** 13](#_Toc26708074)

[**4.2 用例2——司机** 15](#_Toc26708075)

[**4.3 用例 3——业务监测人员** 16](#_Toc26708076)

[**4.4用例4——系统管理人员** 17](#_Toc26708077)

[**5.逻辑视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 17](#_Toc26708078)

[**5.1体系结构概述 —— 包和子系统分层[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 18](#_Toc26708079)

[**5.1.1 应用层** 19](#_Toc26708080)

[**5.1.2 商务服务层** 19](#_Toc26708081)

[**5.1.3 中间件层** 19](#_Toc26708082)

[**5.1.4 基本重用** 19](#_Toc26708083)

[**5.2体系结构概述 —— 类图关系[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 19](#_Toc26708084)

[**6.流程视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 20](#_Toc26708085)

[**6.1流程[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 21](#_Toc26708086)

[**6.1.1临时打车** 21](#_Toc26708087)

[**6.1.2预约打车** 21](#_Toc26708088)

[**6.1.3司机接单** 21](#_Toc26708089)

[**6.1.4司机抢单** 22](#_Toc26708090)

[**6.1.5路线规划** 22](#_Toc26708091)

[**6.1.6行程分享** 22](#_Toc26708092)

[**6.1.7付费** 22](#_Toc26708093)

[**6.1.8监测业务流程** 22](#_Toc26708094)

[**6.1.9监测行程信息** 22](#_Toc26708095)

[**6.1.10管理数据** 22](#_Toc26708096)

[**6.2设计元素的过程[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 22](#_Toc26708097)

[**6.2.1行程缓存** 23](#_Toc26708098)

[**6.2.2路线缓存** 23](#_Toc26708099)

[**6.3流程模型到设计模型依赖关系[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 23](#_Toc26708100)

[**7.部署视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 24](#_Toc26708101)

[**7.1手机终端[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 25](#_Toc26708102)

[**7.2系统后台[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 25](#_Toc26708103)

[**7.3各类用户[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 25](#_Toc26708104)

[**7.4数据库[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 25](#_Toc26708105)

[**7.5第三方系统[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 26](#_Toc26708106)

[**8.结构化视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 26](#_Toc26708107)

[**8.1概念级体系结构视图** 26](#_Toc26708108)

[**8.2模块级体系结构视图** 26](#_Toc26708109)

[**8.3运行级体系结构视图** 27](#_Toc26708110)

[**9.尺寸和性能[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 27](#_Toc26708111)

[**10.质量分析及评价[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)** 28](#_Toc26708112)

[**10.1质量要求** 28](#_Toc26708113)

[**10.1.1可靠性** 28](#_Toc26708114)

[**10.1.2可用性** 28](#_Toc26708115)

[**10.1.3 安全性** 29](#_Toc26708116)

[**10.1.4 可维护性** 29](#_Toc26708117)

[**10.1.5 可移植性** 29](#_Toc26708118)

[**10.1.6可扩展性** 29](#_Toc26708119)

[**10.2场景分析** 30](#_Toc26708120)

[**10.2.1 用例场景** 30](#_Toc26708121)

[**10.2.2 增长性场景** 30](#_Toc26708122)

[**10.2.3 探索性场景** 30](#_Toc26708123)

[**10.3 原型分析** 30](#_Toc26708124)

[**10.4 风险** 31](#_Toc26708125)

**软件架构文档**

**1.****简介[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

**1.1目的[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

本体系结构设计文档是在对某某打车软件系统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，旨在阐述某某打车软件系统的总体结构，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。从而设计一个通用的体系结构，能够支持滴滴打车、美团打车、高德打车等软件的开发，以降低软件开发商的工作量，软件系统维护商的工作量。

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[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

本软件体系结构文档提供了智慧打车软件系统的体系结构概述。智慧打车软件系统正在开发。

系统名称：某某打车软件系统

文档内容：本体系结构设计文档概括地描述了智慧打车软件系统的主要功能，阐述了系统的总体结构，说明了系统的总体设计策略，给出了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。

应用范围：本软件体系结构设计文档适合于智慧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，以及智慧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[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

表格1-1术语定义表

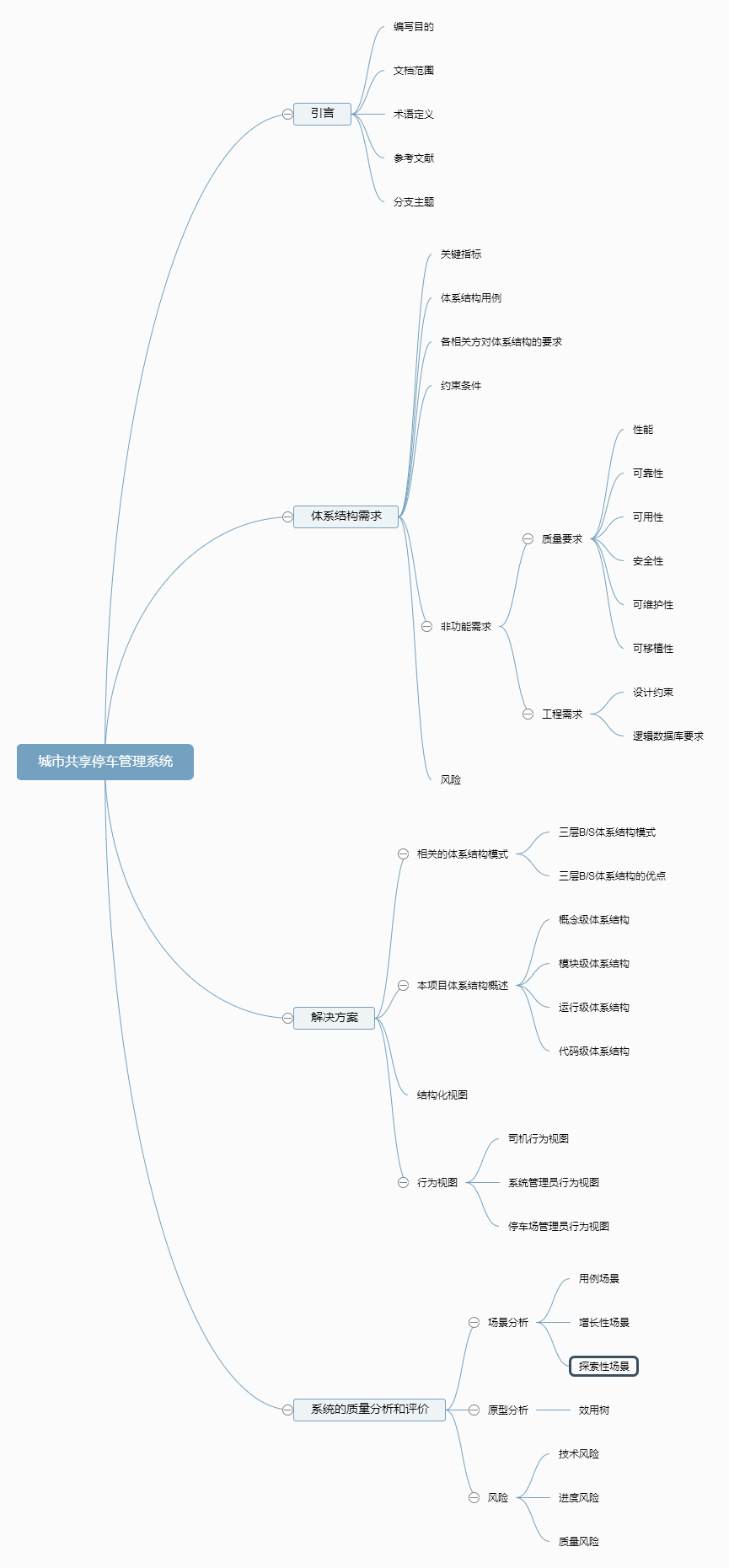
|  |  |
| --- | --- |
| 术语 | 解释 |
| B/S结构 | B/S 结构（Browser/Server，浏览器/服务器模式），是 WEB 兴起后的一种网络结构模式，WEB 浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实 现的核心部分集中到服务器上，简化了系统的开发、维护和使用。 |
| 三层B/S结构 | 三层 B/S 体系结构是在 B/S 结构的基础上，在数据管理层(Server)和用户界面层(Client) 增加了一层结构，称为中间件(Middleware)， 使整个体系结构成为三层。 |
| 概念级体系结构 | 描述系统的主要设计元素和元素之间的关系 |
| 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.反映了对软件代码实现时的期望。 |
| 运行级体系结构 | 描述系统的动态结构。 |
| 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。 |
| 打车软件系统 | 即该软件的名称 |
| 软件 | 为方便叙述，用“软件”来代指智慧打车软件系统。 |
| STSS | 智慧打车软件系统（Smart Taxi Software System，STSS）的简称。 |
| 用户 | 使用软件的所有使用者。 |
| 司机 | 使用软件提供打车服务的司机。 |
| 业务监测员 | 对软件业务进行监测的业务检测人员。 |
| 系统管理员 | 对软件进行管理的系统管理人员。 |
| 接口 | 对软件外部接口，包括第三方支付软件及第三方导航软件等的接口。如微信、支付宝、高德地图等的简称。 |

**1.4****参考文献[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

[1] 王安生. 软件工程化[M]. 北京：清华大学出版社,2015.

## 1.5文档组织结构

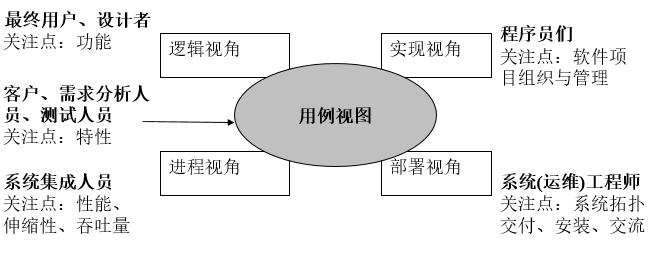
本文档将按图1-1来介绍城市共享停车管理系统的体系结构。



**图1-1 某某网约车系统体系结构图**

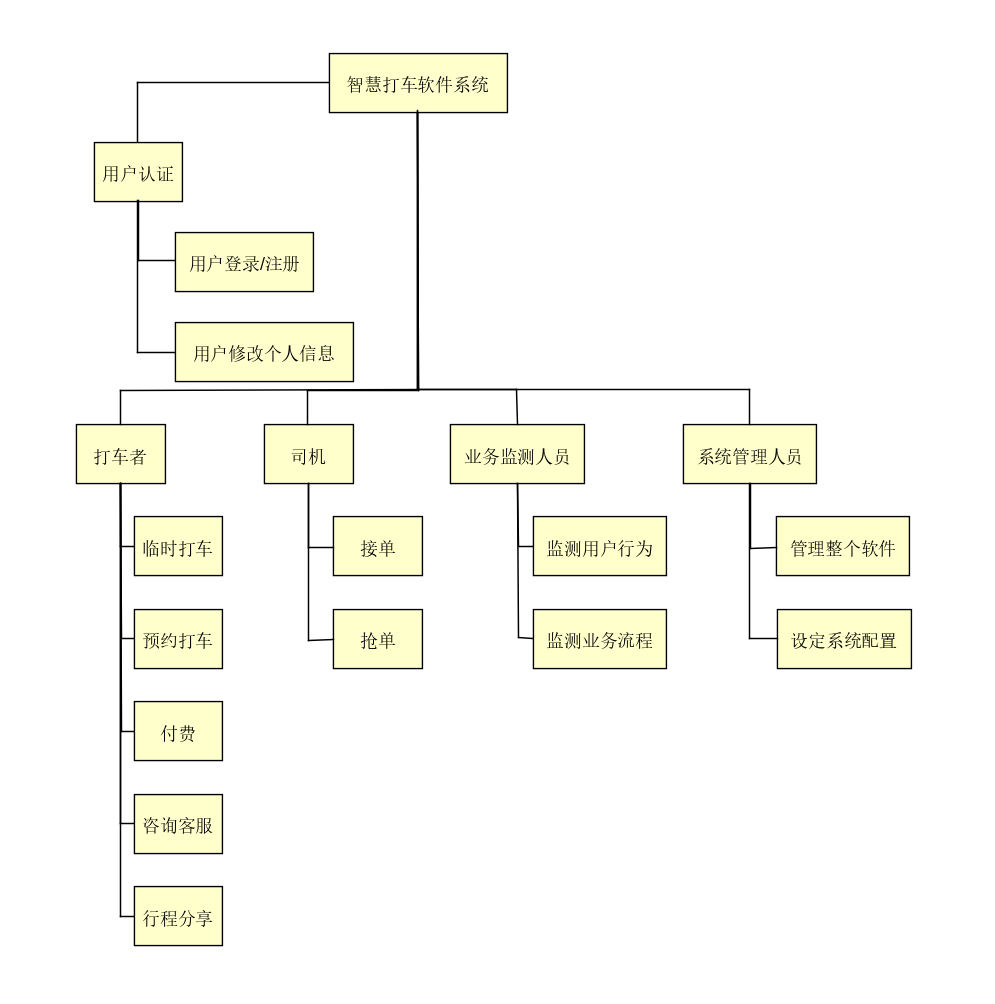
**2.架构表现[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

本打车软件系统结构视角如图所示，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，分析其影响系统的上下文和商业目标情况。本文档中没有介绍单独的实现视图。这些是使用 Rational Rose 开发的基础统一建模语言 （UML） 模型的观点。



**图2-1用例视图**

系统的最终用户和设计者要求系统可以实现需求分析阶段的基本功能，对于网约车打车软件系统，要求系统能满足乘客、司机、业务监测人员、系统管理人员四种不同用户的功能需求。通过限制不同角色的使用权限，实现安全、快速、便捷的打车、抢单、监测用户行为、管理系统等如下图所示功能。节省了用户时间，缓解了城市的打车压力。



**图 2-2基本功能**

用户、分析人员和测试人员要求系统满足需求分析阶段设想的系统特性，要求系统能正确可靠的运行，数据完整，系统可操作性强且用户体验良好。

系统集成人员要求系统满足需求分析阶段提出的性能指标，如系统时间性能和空间性能上的要求。要求系统各个模块和层次之间尽量松耦合以方便各个部件之间的集成。

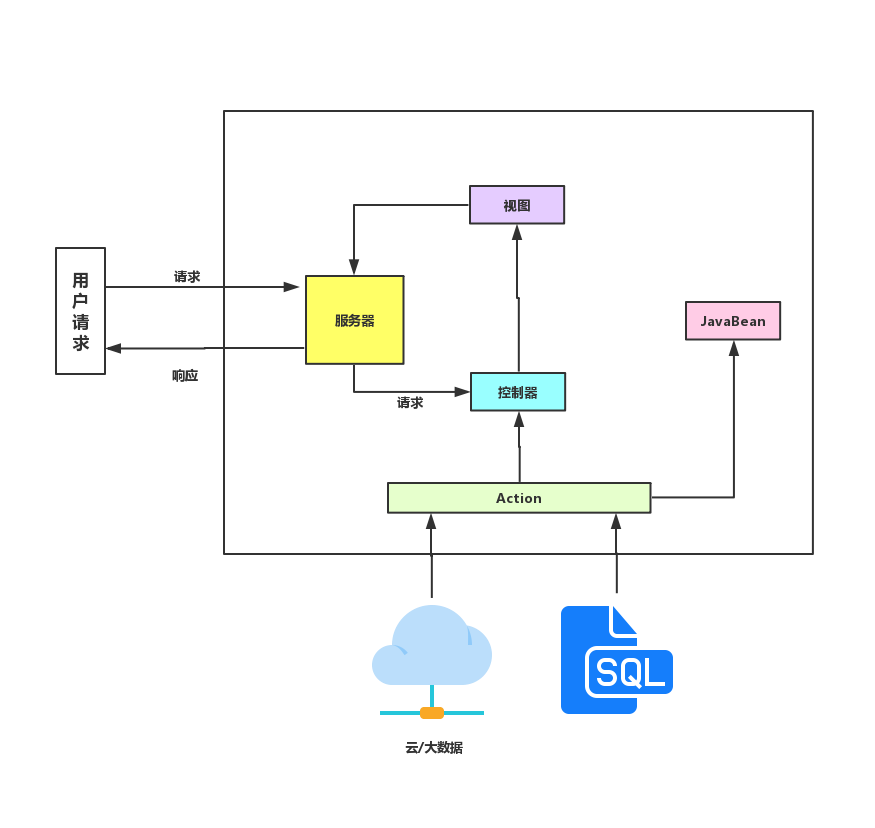
程序员要求系统软件各部分结构合理，方便进行修改、升级、测试和维护，要求系统接口设计合理，便于进行某些模块功能代码的重用，方便被当作模版框架来进行其他系统的二次开发。

系统工程师要求系统在规定时间内完成并交付，并在系统交付时对项目负责人进行相应的培训以便其正确使用系统。

**2.1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。



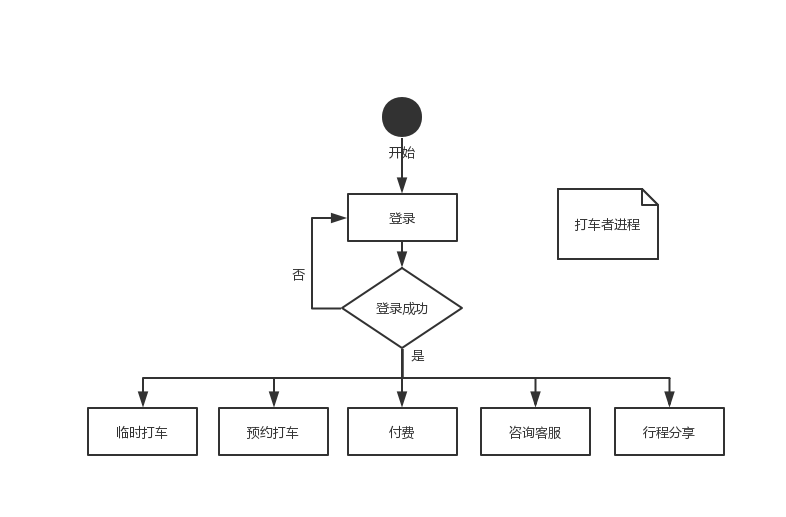
**图 2-3软件逻辑架构示意图**

**2.2 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。

**2.2.1 乘客进程**

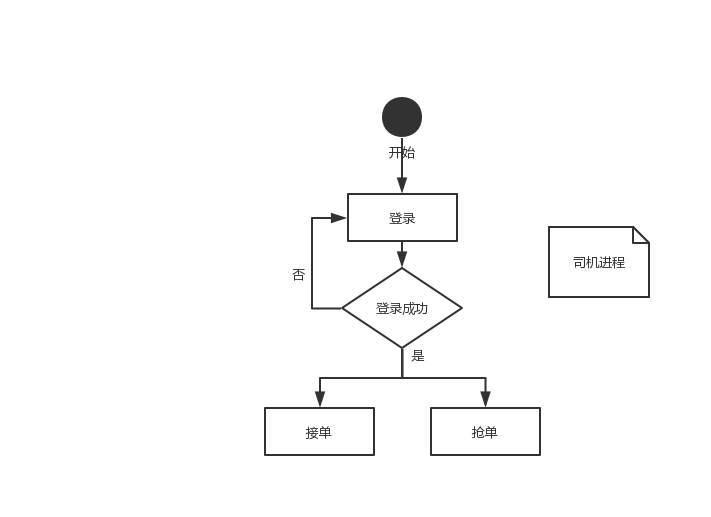
乘客登陆并选择自己的操作，即临时打车、预约打车、付费、行程分享、咨询客服；



**图2- 4打车者进程**

**2.2.2司机进程**

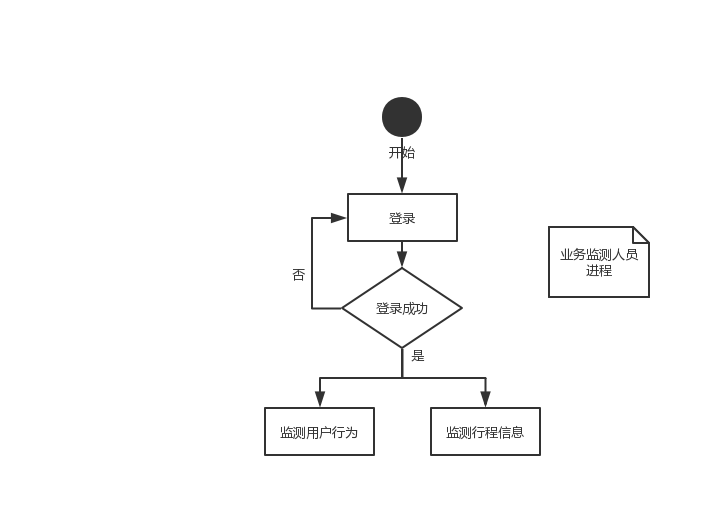
司机登陆并选择自己的操作，即接单、抢单。收费是默认状态下直接到账，在此不做特殊强调。



**图表2- 5司机进程**

**2.2.3 业务监测人员进程**

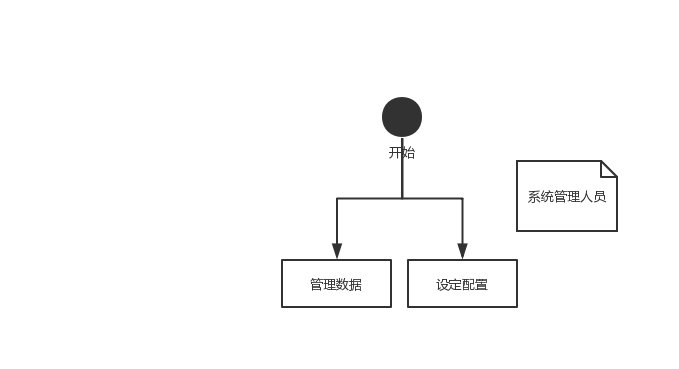
业务监测人员登陆并选择自己的操作，即监测用户行为，监测行程信息



**图2- 6业务监测人员进程**

**2.2.4 系统管理人员**

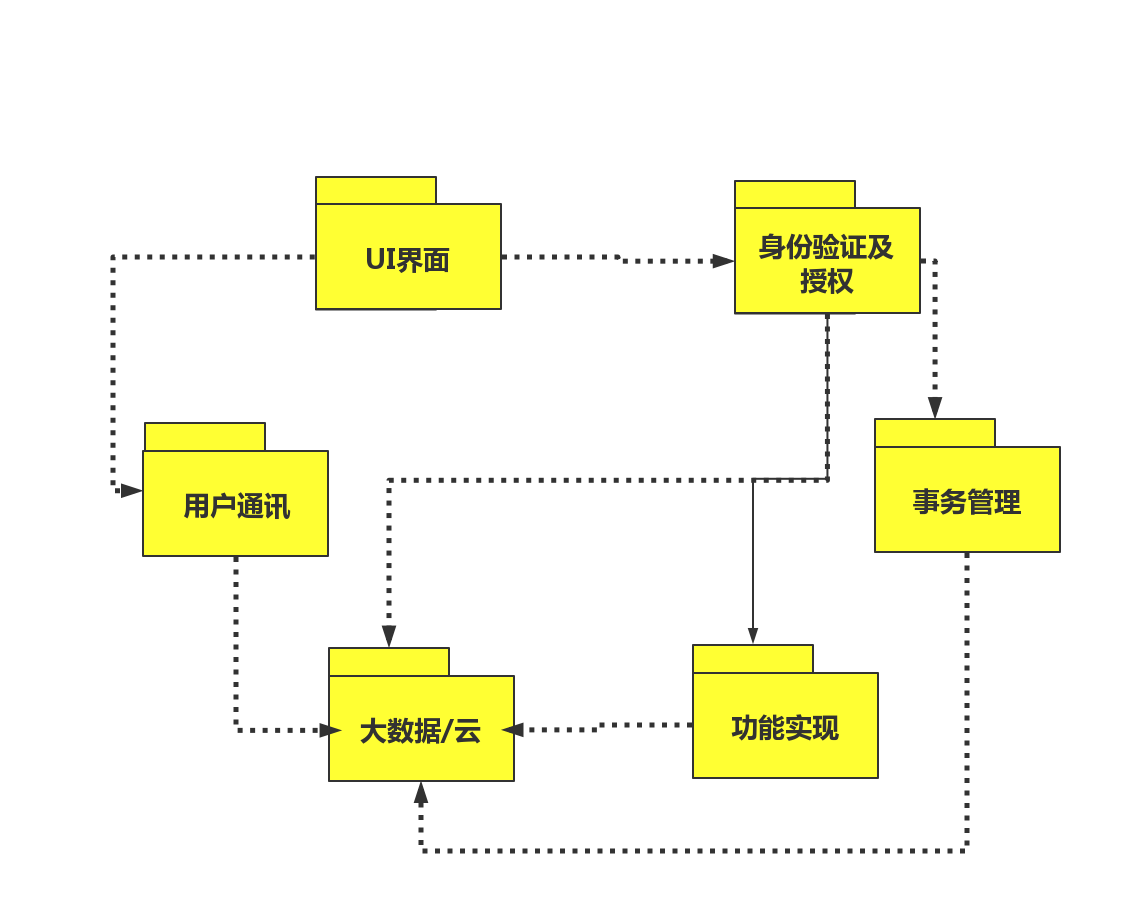
系统管理人员登陆并选择自己的操作，即管理数据、设定配置；其中管理数据包括打车者、司机及业务监测人员和本身系统管理人员的各项数据，配置包括软件系统的各项数据。



**图表2- 7系统管理人员进程**

**2.3 实现视角**

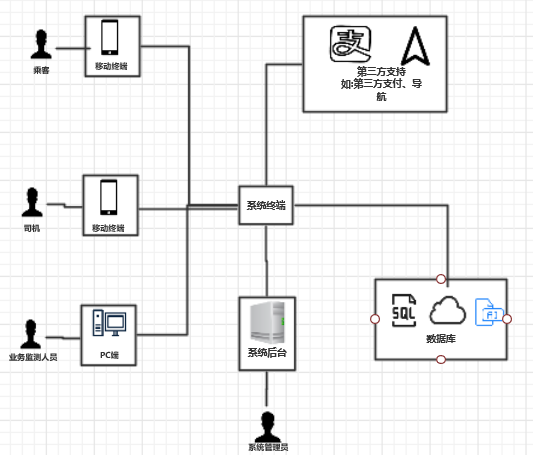
实现视角，又叫开发视图（Development View），描述了在开发环境中软件的静态组织结构，即关注软件开发环境下实际模块的组织，服务于软件编程人员。将软件打包成小的程序块（程序库或子系统），它们可以由一位或几位开发人员来开发。子系统可以组织成分层结构，每个层为上一层提供良好定义的接口。



**图表2- 8开发视图**

**2.4 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2- 9部署视图**

**3.架构目标和约束[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

**3.1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、美团打车、高德打车等软件的开发。

**3.2 设计约束**

1. 软件开发可使用的框架为：Creat系列框架（java框架、c++框架、.net框架、app框架、php框架）
2. 服务器部署于 CentOS系统之上以保证提供稳定可靠的服务。
3. 基于软件可靠性的考虑，软件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。数据存储可实施在云端服务器。
4. 为了软件将来的可扩展性，不应使用具有针对性的需求。整个软件也应尽量减少各模块间的耦合度，提升模块的内聚性。

**3.3 逻辑数据库要求**

本软件将使用数据库对用户信息、司机信息等进行存储。要求如下：

1. 对于来自不同政府部门、第三方服务提供商以及伴随业务产生的大量数据，及时存储，并利用大数据相关技术进行数据挖掘与管理。
2. 使用大型数据库，如Oracle等，可靠性稳定性均较高。便于利用机器学习利用已有数据优化调度算法。

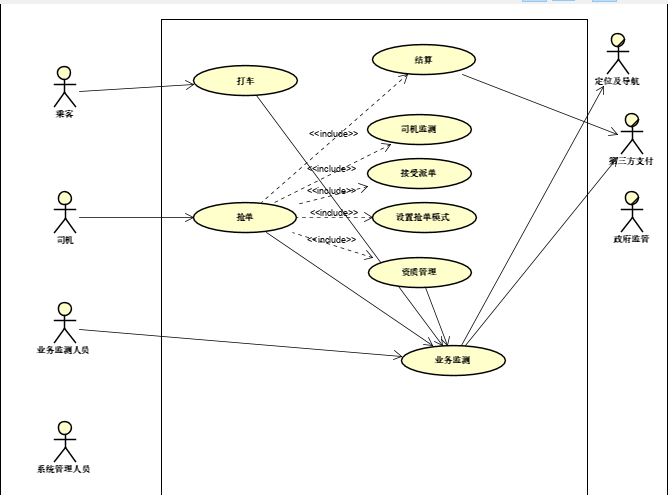
**3.4其他约束**

1. 所有用户必须注册并成功登录后方可使用打车功能。
2. 司机必须经过相关法律法规规定的资格审核方可在平台上进行经营活动。
3. 现有的计费及付款系统必须与第三方/系统自带付费软件进行连接，以支持用户计费及付款。此接口在计费接口规范中定义。
4. 现有的导航规划系统必须与第三方/系统自带导航软件进行连接，以支持用户在形成过程中明确路线。此接口在导航接口规范中定义。
5. 本网约车软件系统必须确保数据完全保护，防止未经授权的访问。所有访问均受用户身份和密码控制的约束。
6. 在开发体系结构时，必须考虑需求文档和补充规范中规定的所有性能和装载要求。
7. 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。

**4.****用例视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

用例图是由参与者，功能用例以及它们之间的关系构成的图，其目的是描述系统功能，通过用例图呈现参与者和他们之间的关系构成的图，就可以更清晰地了解用户对系统、子系统或者各项功能的使用和行为。 本智慧打车软件系统的用户角色包括打车者、司机、业务监测人员、系统管理人员。不同的用户对应着不同的功能，针对这些用户角色，可以画出如下的用例图。

**4.1 用例1——打车者**

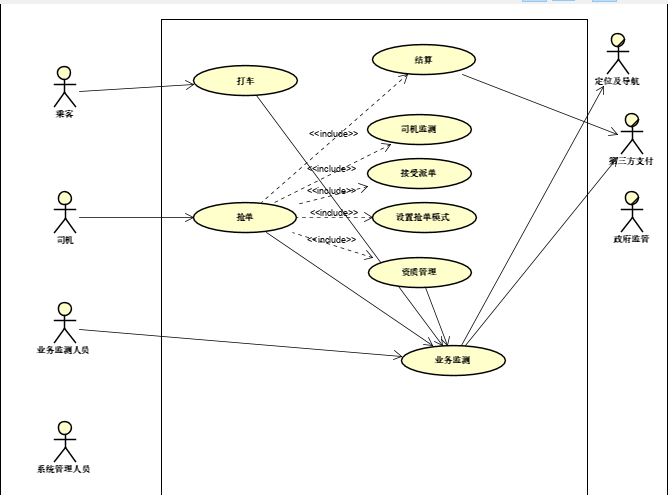


**图4-1打车者用例**

|  |  |  |
| --- | --- | --- |
| ID | 名称 | 描述 |
| FR1 | 临时打车 | 乘客登陆后，选择出发地、目的地后规划出行路径，选择需要的服务类型，生成订单信息并提交给系统，由派车系统匹配司机提供服务。到达目的地，乘客点击确认完成本次业务。 |
| FR2 | 显示实时位置 | 乘客在车上时，向乘客实时显示车辆位置，并提供导航服务。 |
| FR3 | 乘客支付车费 | 乘客确认到达目的地后选择常用的第三方网络支付方法向平台支付车资。 |
| FR4 | 用户认证 | 登录/注册，进行实名认证，修改个人信息，设定个人资料 |
| FR5 | 预约打车 | 打车者设定上车点及下车点，选定时间，系统派单，司机在指定时间前到达指定地点等待用户上车 |
| FR6 | 咨询客服 | 对软件或行程存在的任何困惑都可以咨询客服得到解答 |
| FR7 | 行程分享 | 乘客在分享前要设定好分享对象，将自己此次打车的路线实时分享给指定用户或第三方软件中。指定用户获取当前用户行程，分享完成，返回。 |

**表4-1打车者用例**

**4.2 用例2——司机**

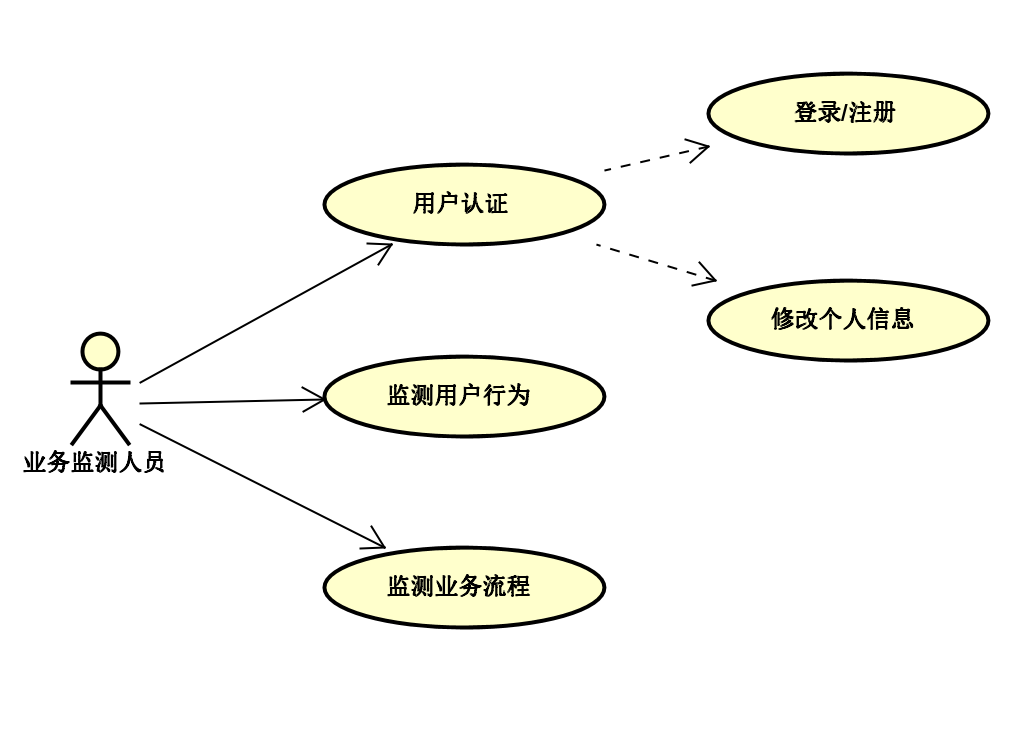


**图 4-2 司机用例**

|  |  |  |
| --- | --- | --- |
| ID | 名称 | 描述 |
| FR1 | 抢单 | 乘客登陆后，选择出发地、目的地后规划出行路径，选择需要的服务类型，生成订单信息并提交给系统，由派车系统匹配司机提供服务。在协议规定的范围内平台可强制要求司机接单。 |
| FR2 | 资质管理 | 司机提交个人驾驶记录、车辆信息、身份信息等信息，平台配合相关政府部门对网约车司机进行资格审核。 |
| FR3 | 司机监测 | 乘客在车上时，向乘客实时显示车辆位置，并提供导航服务。要求司机在每次订单开始前进行人脸识别等操作。 |
| FR4 | 设置抢单模式 | 登录/注册，进行实名认证，修改个人信息，设定个人资料 |
| FR5 | 预约服务 | 司机与乘客通过平台提前约定服务的时间、路线等信息 |
| FR6 | 咨询客服 | 对软件或行程存在的任何困惑都可以咨询客服得到解答 |
| FR7 | 行程分享 | 司机在分享前要设定好分享对象，将自己此次业务的路线实时分享给指定用户或第三方软件中。指定用户获取当前用户行程，分享完成，返回。 |

**表4-2 司机用例**

**4.3 用例 3——业务监测人员**

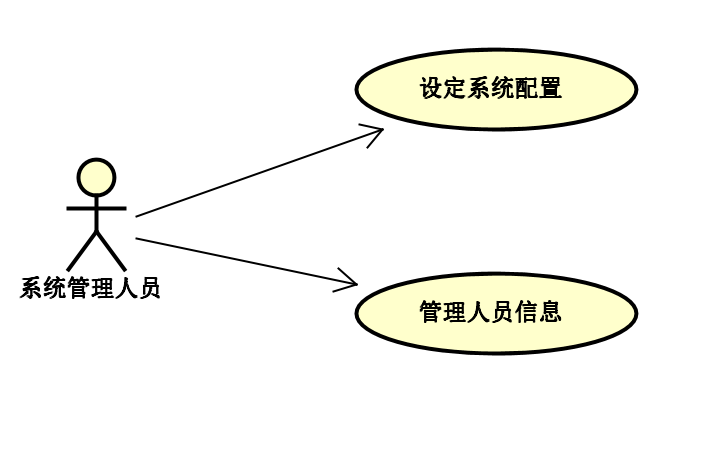


**图 4-3业务监测人员用例**

|  |  |  |
| --- | --- | --- |
| ID | 名称 | 描述 |
| FR1 | 用户认证 | 管理乘客信息， |
| FR2 | 资质审核管理 | 平台配合相关政府部门对网约车司机进行资格审核。 |
| FR3 | 监测用户行为 | 业务监测人员对用户的实时行为进行抽样检测，对全体用户行为进行系统实时监测，明确用户行程路线。根据不同用户的订单时间、订单地点等内容，形成用户画像。 |
| FR4 | 监测业务流程 | 对打车者及司机的各项业务进行实时监测流程规范。针对流程进行监测，观察合理性、规范性及流畅性。 |

**表 4-3业务监测人员用例**

**4.4用例4——系统管理人员**



**图4-4系统管理人员用例**

|  |  |  |
| --- | --- | --- |
| ID | 名称 | 描述 |
| FR1 | 系统管理 | * 管理各类用户信息，包括用户登录/注册及实名认证时获取到的信息及账号权限管理 * 管理软件运营规范 * 针对软件更新迭代进行实时更新，发布新版软件 * 收集用户反馈，整理汇总 |
| FR2 | 系统维护与管理 | 对系统进行日常维护、管理、升级等操作 |

**表4-4系统管理人员用例**

**5.****逻辑视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

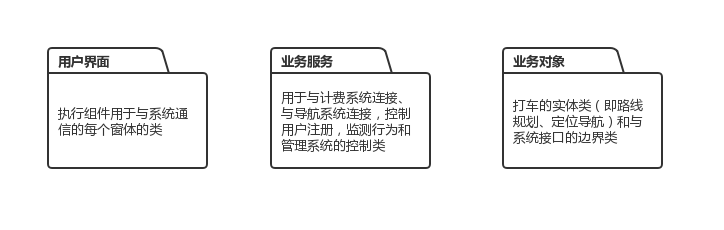
体系结构逻辑视图用来描述最重要的类、服务包和子系统中的组织以及这些子系统在层中的组织。还描述了最重要的用例实现，如体系结构的动态方面。用类图来说明具有体系结构意义的类、子系统、包和层之间的关系。

智慧打车软件系统的逻辑视图由3 个主要包组成：用户界面、业务服务和业务对象。

用户界面包包含执行组件用于与系统通信的每个窗体的类。边界类支持用户登录、打车（包含预约打车等），抢单，接单。

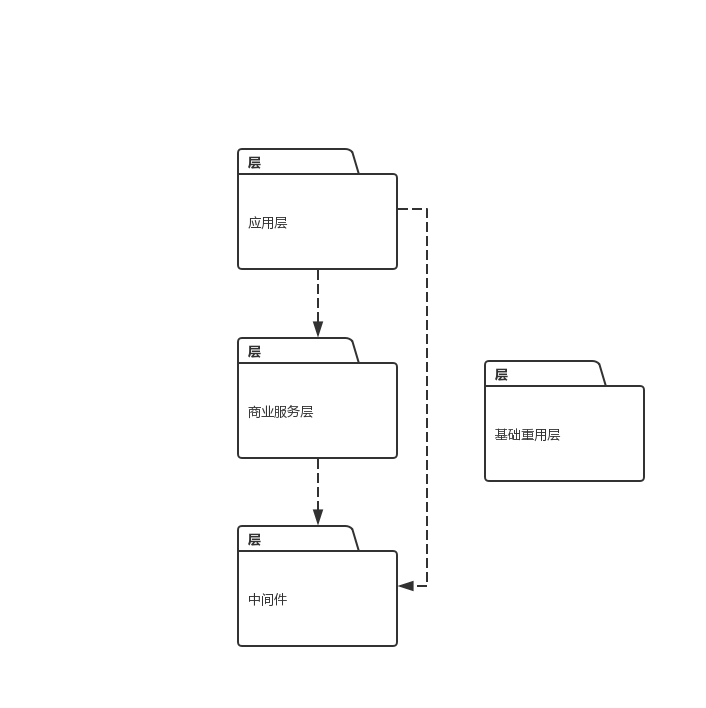
业务服务包包含用于与计费系统连接、与导航系统连接，控制用户注册，监测行为和管理系统的控制类。

业务对象包包含打车的实体类（即路线规划、定位导航）和与系统接口的边界类。



**图5-1 逻辑视图**

**5.1****体系结构概述 —— 包和子系统分层[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**



**图 5-2 包和子系统分层图**

**5.1.1 应用层**

主要实现系统的核心功能，实现包括注册、登陆、身份校验、订单调度、订单持续监测、咨询、付款、编辑用户信息等业务逻辑。联系系统的前端与后端。

此应用程序层具有表示用户看到的应用程序屏幕的所有边界类。此层取决于"进程对象"图层;跨越客户端与中端的分离。

**5.1.2 商务服务层**

业务服务流程层具有表示驱动应用程序行为的用例管理器的所有控制器类。此层表示客户端到中间层边框。业务服务层依赖于"流程对象"层;跨越客户端与中端的分离。

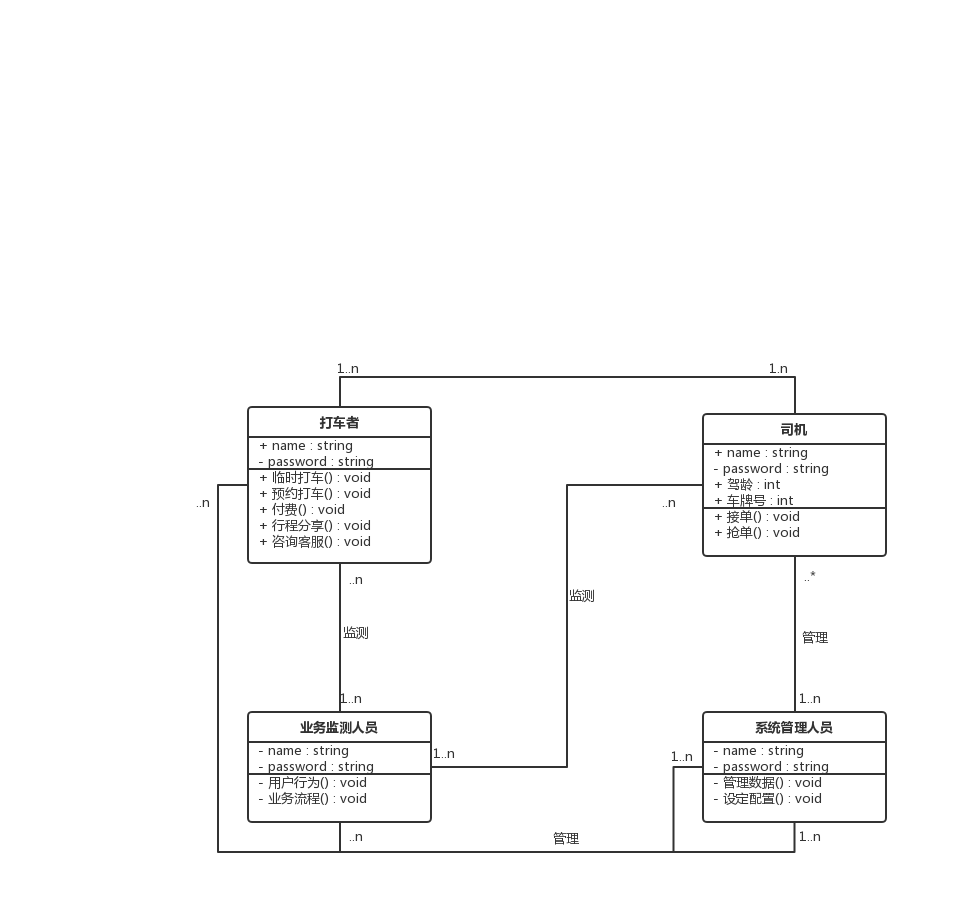
**5.1.3 中间件层**

中间件层支持对关系 DBMS 和 OODBMS 的访问。

**5.1.4 基本重用**

基本重用包包括类以支持列表函数和模式。

**5.2体系结构概述 —— 类图关系[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**



**图 5-3 类图关系**

打车者具备两个属性：

* + - 用户名
    - 密码

打车者具有五个功能：

* + - 临时打车
    - 预约打车
    - 付费
    - 咨询客服
    - 行程分享

司机具备四个属性：

* + - 用户名
    - 密码
    - 驾龄
    - 车牌号

司机具有两个功能：

* + - 抢单
    - 接单

业务监测人员具备两个属性：

* + - 用户名
    - 密码

业务监测人员具有两个功能：

* + - 监测用户行为
    - 监测业务流程

系统管理人员具备两个属性：

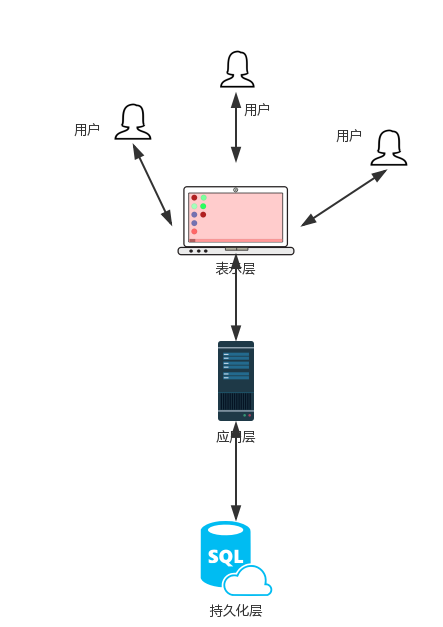
* + - 用户名
    - 密码

系统管理人员具有两个功能：

* + - 管理数据
    - 设定配置

**5.3概念级体系结构[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

概念级的体系结构是从需求工程阶段给出的系统模型直接进化过来的。在概念级体系结构中，表示层展示系统页面，应用层实现系统的主要功能，处理用户请求并返回相关数据，持久化层将用户信息和相关数据保存在数据库中.

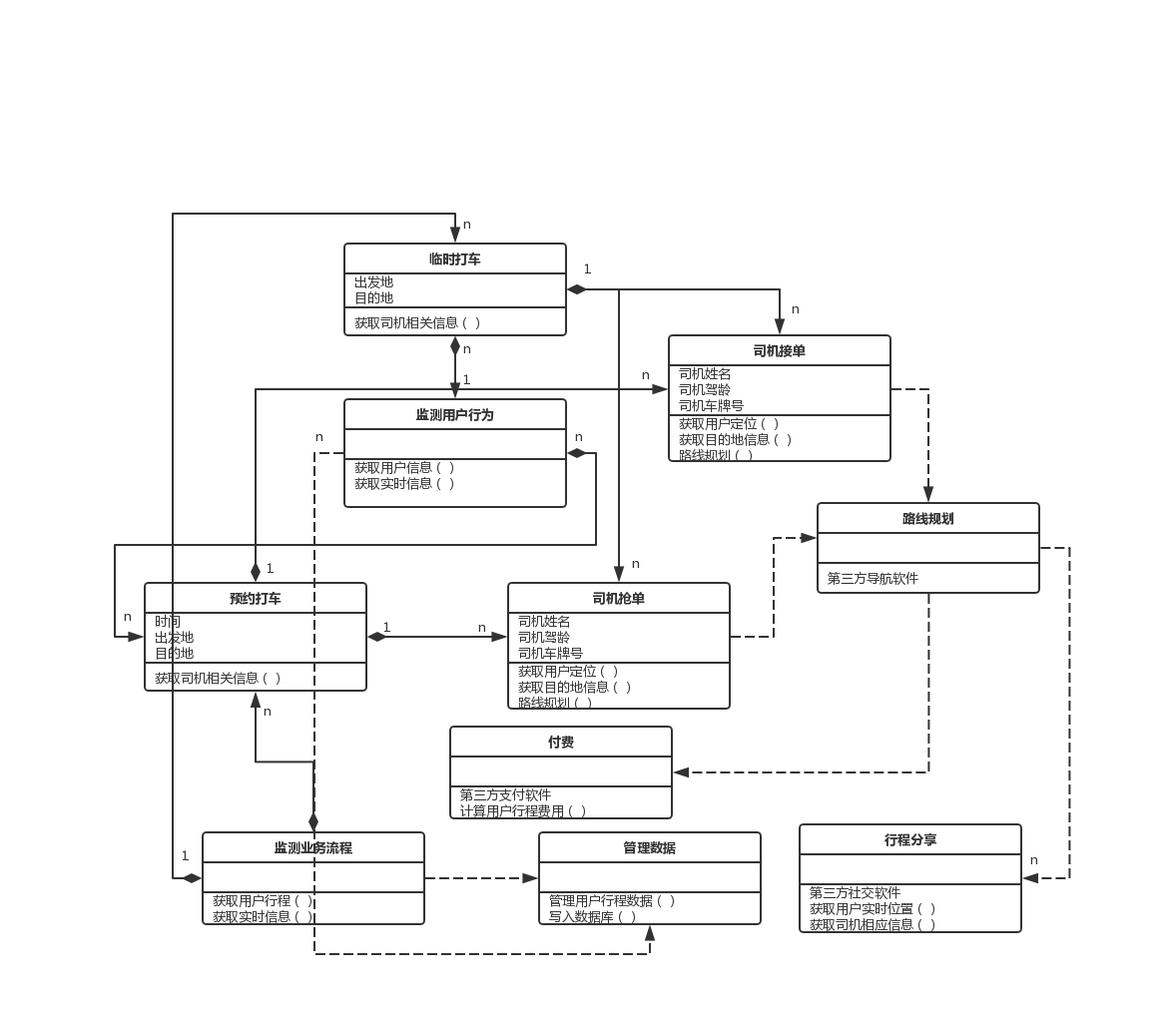


**图5-4概念级体**

**6.****流程视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

体系结构的流程视图描述系统执行中涉及的任务（进程和线程）及其交互和配置。还描述了对象和类对任务的分配。流程模型说明了可执行组织的执行过程。

**6.1****流程[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**



**图 6-1 流程图**

**6.1.1临时打车**

1. 可以选择网约车/出租车；
2. 输入出发地和目的地；
3. 等待接单
4. 确认出行

**6.1.2预约打车**

1. 打开首页，选择出行业务，网约车/出租车/专车/顺风车；
2. 输入出发地和目的地；
3. 选择预约车辆时间；
4. 确认呼叫预约行程

**6.1.3司机接单**

系统分配给司机用户，司机获取用户出发地、目的地，第三方软件提供路线规划及导航功能。

**6.1.4司机抢单**

司机通过系统公布的用户信息，实时抢单，获取用户出发地、目的地，第三方软件提供路线规划及导航功能。

**6.1.5路线规划**

司机接单后由第三方软件根据用户出发地、目的地及实时路况信息进行路线规划，实时反馈给司机及打车者。路线信息会被记录到数据库中。

**6.1.6行程分享**

当打车者的打车申请被接单，打车者顺利乘车后，系统将打车信息（出发地、目的地、规划路线、司机车牌号）通过第三方社交软件发送给打车者预先设定的分享者，分享者获得打车者行程相关信息。

**6.1.7付费出行费用**

此过程系统通过第三方路线规划计算出价格，反馈给打车者及司机，打车者通过第三方支付软件进行支付。

**6.1.8监测业务流程**

业务监测人员实时监测软件使用过程中的问题，并实时反馈。

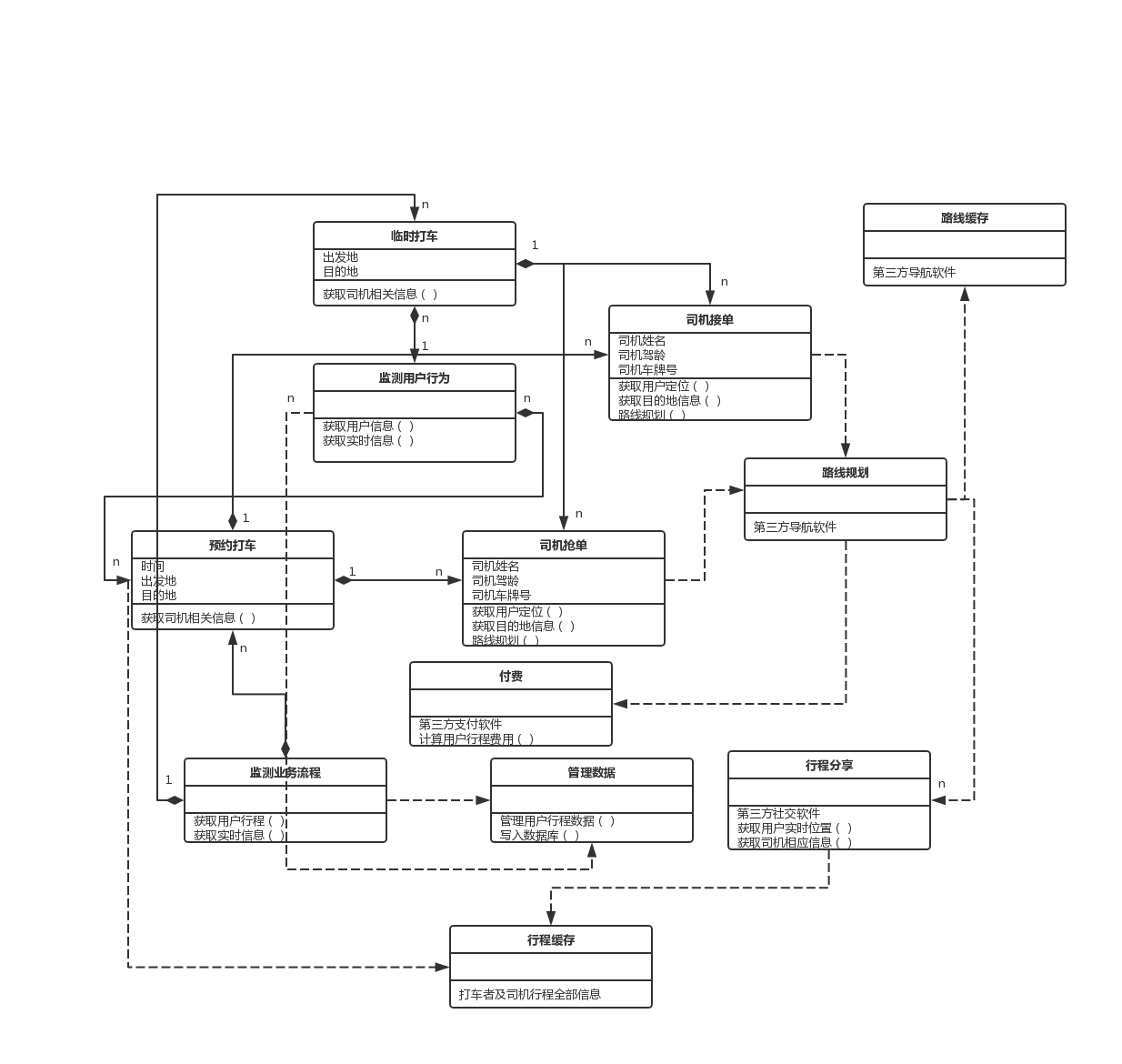
**6.1.9监测行程信息**

业务监测人员针对打车者打车行程相关信息进行实时监控，系统实时监控，业务监测人员实施抽样监控，监测用户打车过程中的行程相关信息。

**6.1.10管理数据**

系统管理员将软件内所有数据写入数据库进行管理，与软件进行通信以达到目的地。

**6.2****设计元素的过程[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**



**图 6-2 流程图系化图**

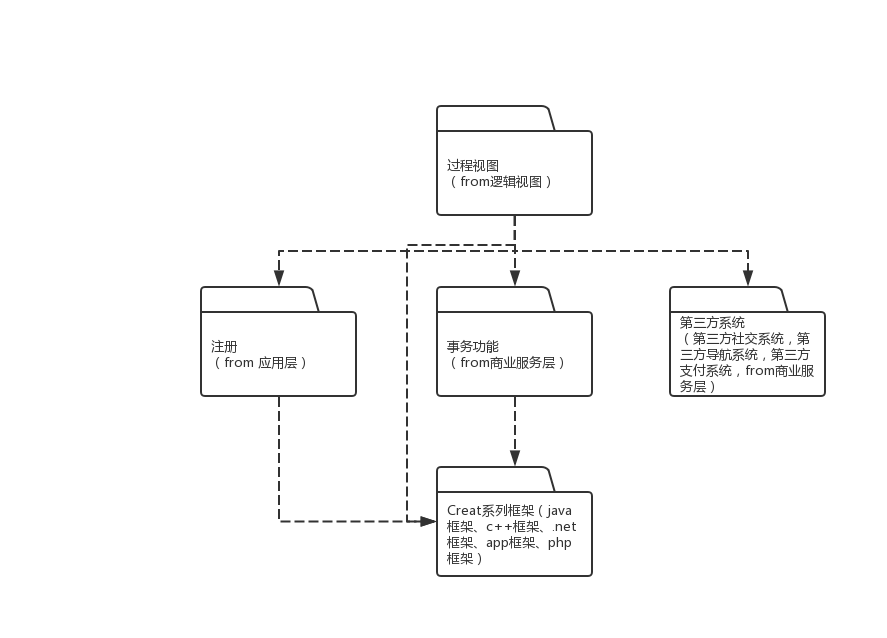
**6.2.1行程缓存**

行程缓存线程用于异步从用户处获取行程全部信息以便实时监测。

**6.2.2****路线缓存**

路线缓存用于异步从第三方导航软件中获取路线相关信息。

**6.3****流程模型到设计模型依赖关系[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

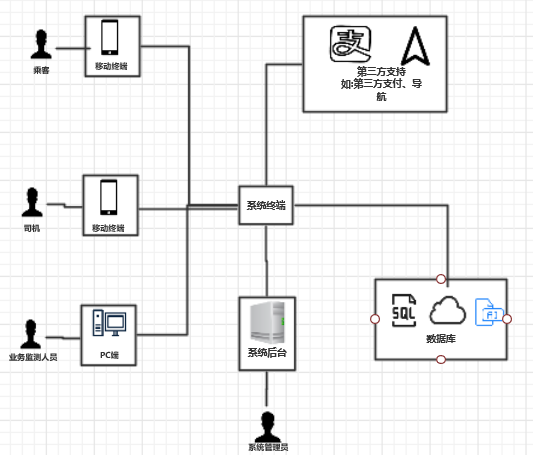


**图 6-3 依赖关系**

**7.****部署视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

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 6-4 部署视图**

**7.1手机终端[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

打车者、司机、业务监测人员通过使用移动终端或PC端访问系统后台内容。

**7.2系统后台[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

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。

**7.3各类用户[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

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。

**7.4数据库[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

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。

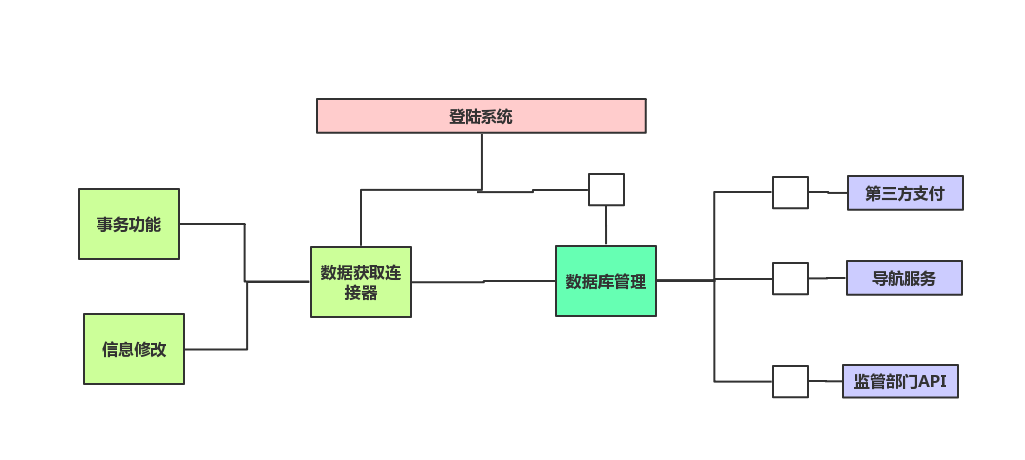
**7.5第三方系统[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

第三方系统包括第三方支付系统、第三方导航系统以及由政府部门提供的API接口，根据不同功能进行不同权限系统的访问。

**8.结构化视图[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

**8.1概念级体系结构视图**

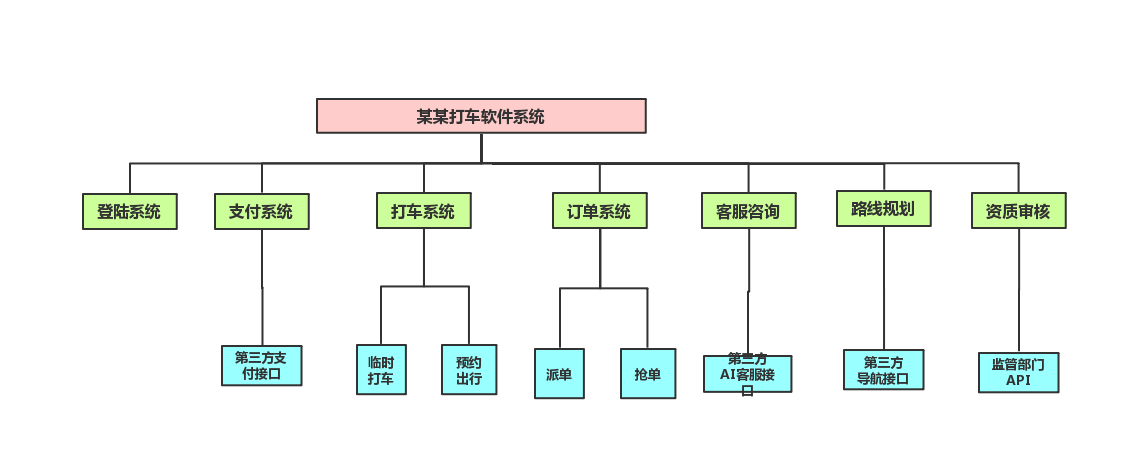
概念级的体系结构是从需求工程阶段给出的系统模型直接进化过来的。在概念级体系结构中，表示层展示系统页面，应用层实现系统的主要功能，处理用户请求并返回相关数据，持久化层将用户信息和相关数据保存在数据库中，概念级体系结构的示意图如下图所示。



**图 8-1 概念级体系结构视图**

**8.2模块级体系结构视图**

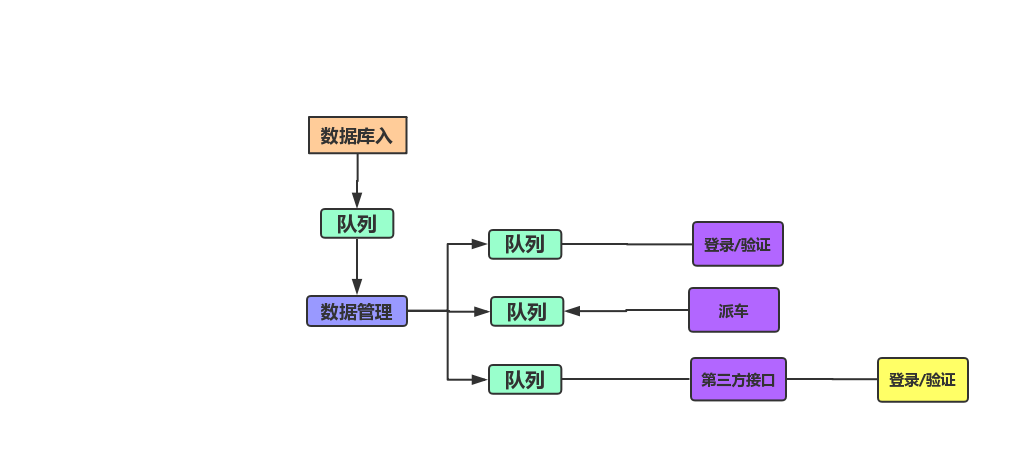
系统的模块化设计是系统进行复用的关键。模块级体系结构反应了对软件代码实现时的期望。特别是对于程序规模较大的系统。下图是按系统的层次进行划分的系统模块化表示。体系结构的层次表达了每层的向上一个层面提供的功能和接口，以及需要使用下一层的功能。



**图 8-2 模块级体系结构视图**

**8.3运行级体系结构视图**

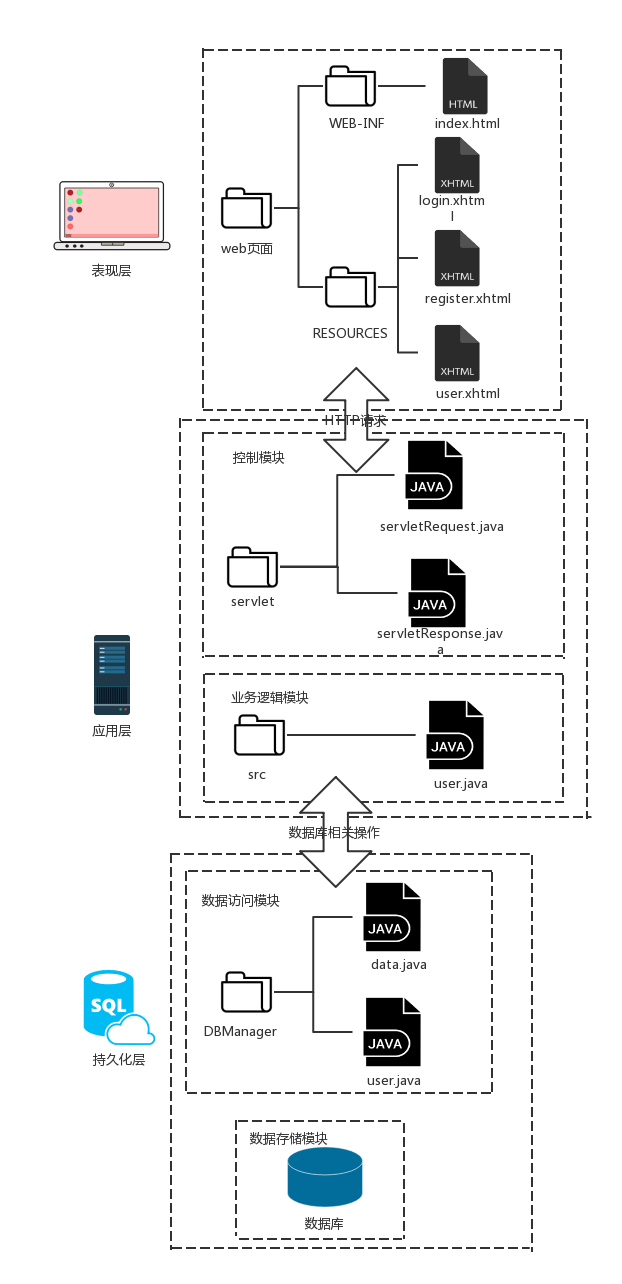
运行级的体系结构用来描述系统的动态结构，用运行级体系结构才能很好地进行性能分析、监视和调整，以及调试和服务维护。本网约车系统的运行级体系结构如图下图所示。



**图 8-3 运行级体系结构示图**

**8.4代码级体系结构视图**

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。本网约车系统的代码级体系结构如下图所示。



**图8-4代码级体系结构示意图**

**9.****尺寸和性能[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

本软件存在大量潜在用户，预计总用户数量在百万级别。高峰时段不同地区同时使用人数可达上万人。因此，本软件对并发性性能要求较高。

客户端软件整体大小应小于500MB。

**时间性能：**

*<<该系统拟定在大中城市分别建立数据中心为周边地区提供服务，本小节中系统指一个这样的子数据中心>>*

软件执行速度

* 1. 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超过0.5秒。

* 1. 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： 
     1. 系统应当可以获取用户的地理位置及周边位置信息。一个流程的响应速度最佳为 5 s,平均为 10 s，不超过 20 s。系统应当可以同时接受10000人同时操作。
     2. 系统应当可以获取周围空闲车辆信息。一个流程的响应速度最佳为 5 s,平均为 8 s，不超过 15 s。系统应当可以同时接受10000人同时操作。
     3. 系统应当能够支持更新周围车辆信息及用户位置信息。一个流程的响应速度最佳为 3 s,平均为 6 s，不超过 10 s。系统应当可以同时接受10000人同时操作。
     4. 系统应当能够计算出发地到目的地最佳路线信息，实时获取路况信息，避免拥堵。一个流程的响应速度最佳为 4 s,平均为 8 s，不超过 15 s。系统应 当可以同时接受10000人同时操作。
     5. 系统应当能够赋予司机抢单权限。一个流程的响应速度最佳为 3 s,平均为 5 s，不超过 10 s。系统应当可以同时接受10000人同时操作。

**空间性能：**

1. 系统应当能够获取用户地理位置及周边位置信息，每次操作占用内存不超过 20MB。
2. 系统应当能够获取周围空闲车辆信息，每次操作占用内存不超过 30MB。
3. 系统应当能够更新周围车辆信息及用户位置信息，每次操作占用内存不超过 30MB。
4. 系统应当能够计算出发地到目的地最佳路线信息，实时获取路况信息，每次操作占用内存不超过 80MB。
5. 系统应当能够支持司机抢单信息，每次操作占用内存不超过20MB。

**10.****质量分析及评价[top](http://www.ts.mah.se/RUP/wyliecollegeexample/courseregistrationproject/artifacts/analysisndesign/sadoc.htm#Toc)**

**10.1质量要求**

**10.1.1可靠性**

为了保障软件能在较长时间下稳定运行，同时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，软件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。由于在处理用户地理位置、司机地理位置、行程路线规划、抢单等信息时，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。保障在高峰时段用户人数陡增的软件稳定性问题。

（1）系统测试时间至少为100小时，测试参与用户至少为5000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。

（2）系统具有一定的容错能力和故障恢复能力，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，但不会导致系统崩溃无法使用。

（3）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）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。同时应该兼顾企业进行大数据分析的业务需求。

（5）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。

**10.1.2可用性**

该软件的可使用性体现在它可以支持多操作系统运行。同时该软件应该具备容易操作的功能。本软件的设计应建立在用户渴望软件简洁易上手的假设之上。

（1）系统支持在多种操作系统和多种平台、操作系统下运行和使用。

（2）系统界面美观，简洁明了，有较好的用户体验，能吸引使用者。

（3）本系统易于使用。新用户在初次使用系统时，系统会自动显示操作指南指导用户使用系统，用户在不超过3分钟的学习时间之内便可以明白整个系统的基本功能并能熟使用系统。对于司机的资格审核严格按照相关法律法规的要求的前提下尽可能实现无人化审核，将资格审核时间限制在24h之内。

（4）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。

能要便于用户理解，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，可参考目前大部分软件的简洁设计。

**10.1.3 安全性**

本系统的安全性十分重要。为了保证系统的安全性，在这里提出如下要求：

1. 通常来讲，实际使用的应用软件，必须具备相应的安全性能。该软件各级用户有各自的权限设置，例如用户可以查看司机的车牌号、驾龄等信息。司机可以获取用户的手机号码、出发地目的地等信息。
2. 该软件应该通过设置防火墙确保数据传输的安全。使用可靠的加密存储方式来保证软件的操作安全，确保软件在一个相对安全可靠的环境下运行。
3. 软件应保证用户信息不泄露，软件系统配置文件和数据库存储文件应当进行加密处理。
4. 软件应保证不会因恶意攻击而崩溃，在开发过程不存在明显漏洞。
5. 软件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. 严格保证未经认证的用户不能访问到关键信息。

除此之外，对实现方面也提出如下要求：

1. 尽量避免在业务代码中直接使用SQL访问数据库，应当使用ORM来代替。
2. 多次登陆失败之后再次登陆要求输入验证码。
3. 系统对不同用户的权限进行控制，且司机用户使用系统前应先使用身份证信息、驾驶证信息、车辆信息等进行资格审核。业务监测员账号、系统管理员账号进行系统的登录，不同的类型的用户具有不同的系统使用权限。
4. 系统对于重要数据进行数据加密。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。
5. 系统数据定期备份。为了本网约车系统的长远考虑，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。防止系统出错意外崩溃时数据丢失。
6. 归档记录与日志记录。在删除用户前由系统管理员归档相关用户的信息，以用于可能的查询。除此之外，在系统发生故障后，也会由系统管理员归档相应的系统日志信息，以便于之后研究系统发生故障的根源。

**10.1.4** **可维护性**

软件的可维护性是指改进软件的难易程度。该软件的结构、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。

同时，软件代码应当遵从面向对象设计原则，设计上要做到易扩展、易维护，以减少以后的维护和二次开发所需的成本，保证代码的质量。

1. 系统的代码和文档结构应该保证清晰易读，便于在系统出现错误之后程序员能够进行快速的恢复抢修，并且能准确的找出系统错误的原因。
2. 利用稳定可靠的的软件开发技术和工具，提高可维护性。
3. 在软件开发每个阶段结束前的技术审查和管理复审中，着重对可维护性进行审查。
4. 选择可维护的程序设计语言。
5. 改进程序文档。文档是影响软件可维护性的重要因素，好的文档是建立可维护性的基本条件。在软件维护阶段，利用历史文档可以大大简化维护工作。历史文档有3种：系统开发日志、错误记载、系统维护日志。
6. 降低系统耦合度，便于完成后期系统测试。
7. 系统开发完毕后提供必要的操作维护手册和技术手册。

**10.1.5 可移植性**

1)系统能市面上常用的操作系统环境下（主要为IOS、Android端）方便地安装和使用，不会因为环境的不同而导致系统崩溃或者系统功能缺失。

2)使用标准库函数，并且把它们和[ANSI](https://baike.baidu.com/item/ANSI/10401952)/ISO C标准中定义的头文件放在一起使用。

3)尽可能使程序适用于所有的[编译程序](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E7%A8%8B%E5%BA%8F/8290180)，并把不可移植的代码分离出来。

4)系统接口设计合理，便于进行某些模块功能代码的重用，方便被当作模板框架来进行相应其他系统的二次开发。

5)系统具备良好的共存性，能够和其他软件共存于一个平台上，不存在冲突。

6)系统具有良好的可替换性，能够容易地卸载，也易被高版本的系统替换。

7)系统对于不同版本的操作系统具有较好的向下兼容性

**10.1.6可扩展性**

为了满足不同发展的客户需求和业务需求，软件安装后，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性，便于日后系统的升级和修改。

在上述前提下，考虑需求模型能否被拓展应用到共享单车、共享汽车上，实现需求模型的再应用，即可拓展性。本体系结构可以很好的适用于其他应用场景，可扩展性较强。其中数据库链接、第三方接口可作为公共部件进行链接开源，从而获取收益。

**10.2场景分析**

**10.2.1 用例场景**

用例场景是从使用者的角度出发，描述用户期望的与整个运行系统的交互。

|  |  |
| --- | --- |
| 场景-1 | * 乘客希望在查找附近空闲车辆时，可以根据距离或者收费模式进行排序。 * 该场景代表了用户期望系统易于使用，即易用性。 |
| 场景-2 | * 当系统发生故障时，缓存系统要能在1秒钟内从一个处理器切换到另一个处理器。 * 系统测试时间至少为100小时，测试参与用户至少为10000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。 * 系统具有一定的容错能力和故障恢复能力，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，但不会导致系统崩溃无法使用。 * 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。 * 该场景代表了系统可靠性要求。 |
| 场景-3 | * 系统中的数据得到更新后，系统的其他使用者页面的数据要在10秒内得到更新。 * 5000个用户能够并发登陆系统，且此时系统产生的延迟不超过15秒。 * 对用户的查询操作，系统的响应时间不超过5秒。 * 该场景代表了系统的性能要求。 |
| 场景-4 | * 用户登录系统后，打车者可以随时查询周边车辆信息及交通实时状况，司机可以进行实时接单抢单，业务监测人员可以实时监测软件运行情况及抽样监测用户行程。该场景代表了用户期待系统易于使用，即易用性； |
| 场景-5 | * 发生数据异常时，系统要通知系统管理员，并提示修改。该场景代表了用户期望的可靠性； |
| 场景-6 | * 用户通过手机终端使用系统，系统管理员通过电脑端使用系统。该场景代表了用户期望的可使用性； |
| 场景-7 | * 用户在登录系统时，期望填写验证码防止账号密码泄漏。该场景代表了用户期望的可靠性； |
| 场景-8 | * 用户通过手机终端 获得数据报告的时间小于 10 秒。该场景代表了系统的性能要求； |

**表 10-1用例场景**

**10.2.2 增长性场景**

|  |  |
| --- | --- |
| 场景-1 | * 可以增加新的数据服务器，降低远程用户的访问时间。 |
| 场景-2 | * 通过扩充现有数据库表的规模，把查订单匹配时间降低到平均15S、之内。 |
| 场景-3 | * 通过购买更多硬件设备建设更多分布式数据仓库，来提升系统数据库的并发度。 |
| 场景-4 | * 增设app客户端的开发，方便使用不同同平台、不同硬件，不同操作系统用户的使用习惯。 |

**表 10-2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学生成绩管理系统的修改给出更实际的需求。

|  |  |
| --- | --- |
| 场景-1 | * 用户不仅能在PC端用浏览器登录系统并进行相关操作，也能在移动端浏览器或APP上对系统进行操作。 |
| 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 | * 当司机、打车者10000 人同时使用该系统时，系统可承受并且反应时间控制在 5 秒内 |
| 场景-5 | * 对实时路况上传信息进行分析，及时更新路线和道路状况，向用户展示实时可靠的信息； |
| 场景-6 | * 将系统做成一个微信小程序，方便用户更快捷方便使用。 |

**表 10-3 探索性场景**

**10.3 原型分析**

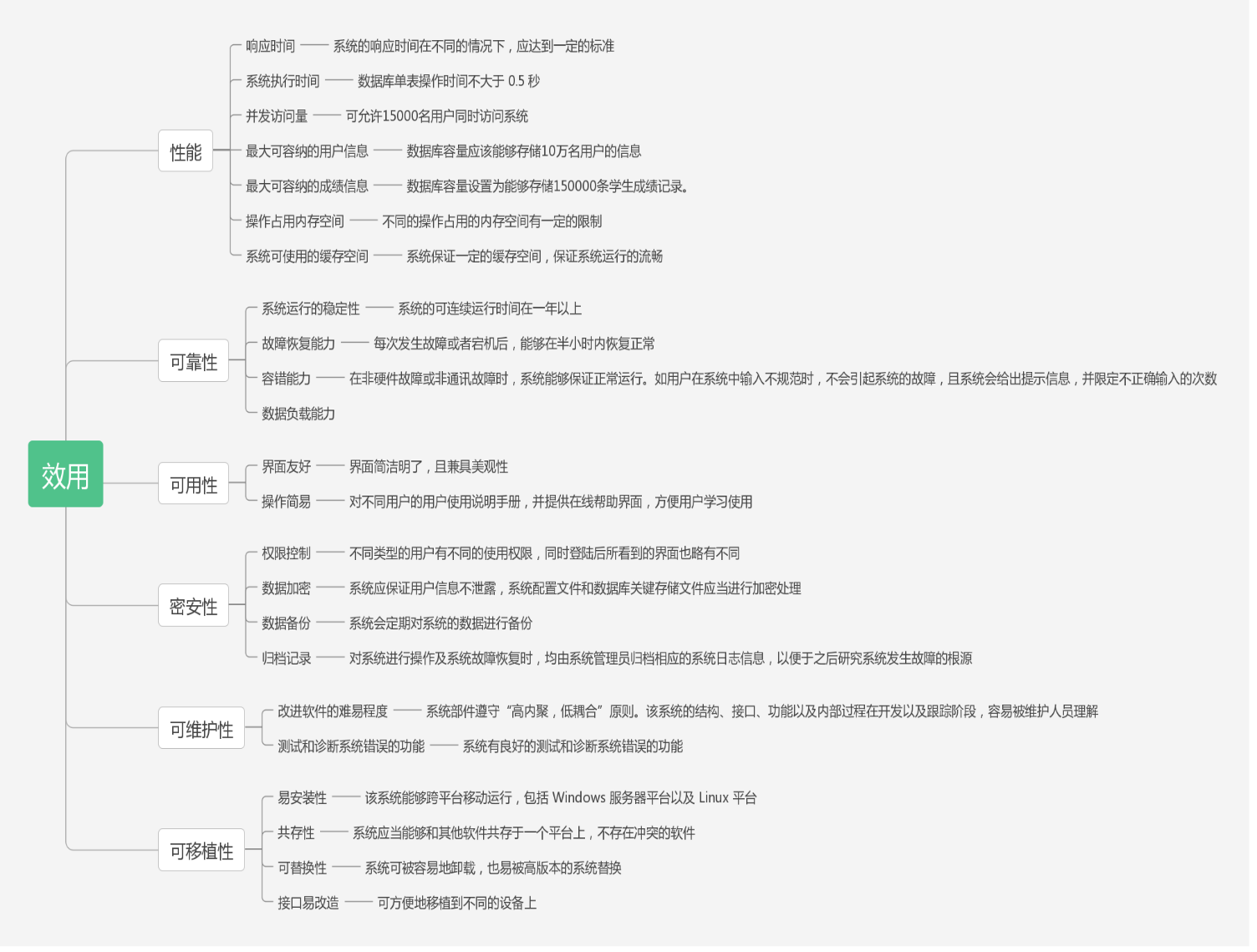
**10.3.1 原型法需求分析**

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： 智慧打车软件系统的用户可分为打车者、司机、业务监测人员、系统管理员。其中打车者的需求是能够安全登录账户，进行打车与支付车费，最好还可以进行咨询客服及行程分享。司机的需求是能够安全登录账户，并尽可能多的接单抢单， 通过规划好的路线将乘客送往目的地。业务监测人员的需求是能够安全登录账户，对软件的运行及用户的行程进行实时监测。系统管理人员的需求是能够对数据和设备进行管理及更新。
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待评价特性，暂时不考虑系统的安全性、健壮性，建立基本满足打车者、司机、业务监测人员、系统管理人员使用要求的系统；
3. 用户和开发人员共同评价原型： 本步骤要求系统的使用者（打车者、司机、业务监测人员、系统管理人员）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。

**图 10-1某某网约车系统质量效用树**

**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。