Better-Than-Uber 打车软件

体系结构设计文档

2.0

2019.12.9

徐雯

软件学院

2017211854

软件工程导论

2019 秋

修订历史记录

日期	描述	作者	说明
2019/12/5	初步完成文档	徐雯	
2019/12/9	补全部署图，效用树等图片，进行检查	徐雯	

审批历史记录

签名	姓名	名称	日期

目录

1. 简介

1.1 目的

本体系结构设计文档是在对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，旨在阐述打车软件设计的总体结构，包括逻辑设计、物理结构，分析系统的体系结构需求，包括约束条件、设计遵循的标准、非功能性需求，给出体系结构设计的解决方案并分析建模，最后进行体系结构的质量分析和评估。

本文档可供此打车软件的用户以及开发人员和维护人员使用。本打车软件应用的目的是对为打车者和司机设计一个软件，使打车者可以在软件中发出打车订单，司机可以在线接单提供打车服务等，缓解高峰期打车难，收费不透明公正的情况，从而为城市内的司机与打车者提供便利，也方便推进城市交通的数字化与智能化。

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，同时方便软件用户、软件客户、开发人员等各方进行软件项目沟通。

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 范围

1.2.1 系统名称

Better-Than-Uber 打车软件

1.2.2.文档内容

本体系结构设计文档概括地描述了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。文档目的是满足系统的质量以及可信赖性要求，以及系统未来的维护、运行和升级改造等要求。

1.2.3.系统功能

打车软件的系统是一个独立的 APP 的系统。该系统在用户的手机上运行，当用户的手机具有 GPS 导航功能且联网的时候系统可以发挥作用。系统对于不同的使用者提供不同的功能，对于司机，系统提供去往打车者所在地的路线的查询功能和去往目的地的查询功能，同时为司机提供查看附近的打车者的订单并且在线接单的功能，在订单完成之后，司机可以通过软件的支付系统获得相应的报酬。对于打车者，系统提供查看位置价格信息的功能，同时，提供打车的功能，打车者可以通过本软件发车打车请求，在打车完成之后，可以通过本软件的支付系统完成支付打车金额的功能。对于客服，系统提供在线与用户交流并解答其问题的功能。对于系统管理人员，本软件提供审核和管理人员信息和车辆信息的功能，同时，系统还提供对这些信息的修改功能。

1.3 定义，首字母缩写词和缩写

表 1 定义与术语

名称	定义
Better-Than-Uber	本文档负责的打车软件
司机	使用该系统的司机用户，拥有合法驾驶资质与正常的行为能力，同时拥有收款账号，能提供驾车服务并接受打车者支付的款项。
系统管理人员	打车软件系统的管理人员，拥有有效的工作资质与正常的行为能力。
打车者	该软件的打车用户，使用该软件进行查询价格和打车活动。

客服	该软件的客服，为打车者和司机解决问题。
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 2. 缩写语

缩写	名词解释
BTU	Better-Than-Uber 打车软件
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。
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。

1.4 参考

[1] 王安生. 软件工程化[M]. 北京：清华大学出版社,2015.

2. 体系结构表示形式

文档以一系列视图的形式介绍了体系结构。 用例视图，逻辑视图，流程视图和部署视图。 本文档中没有单独的实现视图。 这些是使用 Rational Rose 开发的基础统一建模语言（UML）模型的视图。

3. 架构目标和约束

有一些关键要求和系统约束对体系结构有重要影响。

3.1 设计约束

1）必须与支付系统和导航系统，智能计算系统连接，并且能以较快的速度使用该系统并获得返回结果。 以在用户支付的时候，司机开启导航的时候，用户查询到目的地的估计价格的时候提供相应的功能。
2）所有打车者，司机，客服，系统管理人员功能都必须在 PC 和具有 Internet 连接的，已开启 GPS 的智能手机上都可用。
3) BTU 打车软件系统必须确保完全保护数据免遭未经授权的访问。所有远程访问均受用户标识和密码控制。
4）BTU 打车软件系统将作为客户端-服务器系统实施。客户端部分位于 PC 和智能机上，服务器部分必须在服务器上运行。
5）在开发体系结构时，必须考虑本文件中规定的所有性能和负载要求。
6）系统的设计，发行，甲乙方的合约等过程应该符合相关的法律条文。
7）开发过程中第三方工具的使用应该符合相关法律规定。
8）基于系统的安全和保密性考虑，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，防止信息的意外泄露或被攻击。
9） 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。
10）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。

3.2 业务和项目约束

1）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。严格使用面向对象的开发模式，使用 MVC 框架实现前后端的分离，保证系统代码的清晰。
2）在开发体系结构时，必须考虑本文档中规定的所有性能和负载要求。
3）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。

表 2-4 项目设计约束条件

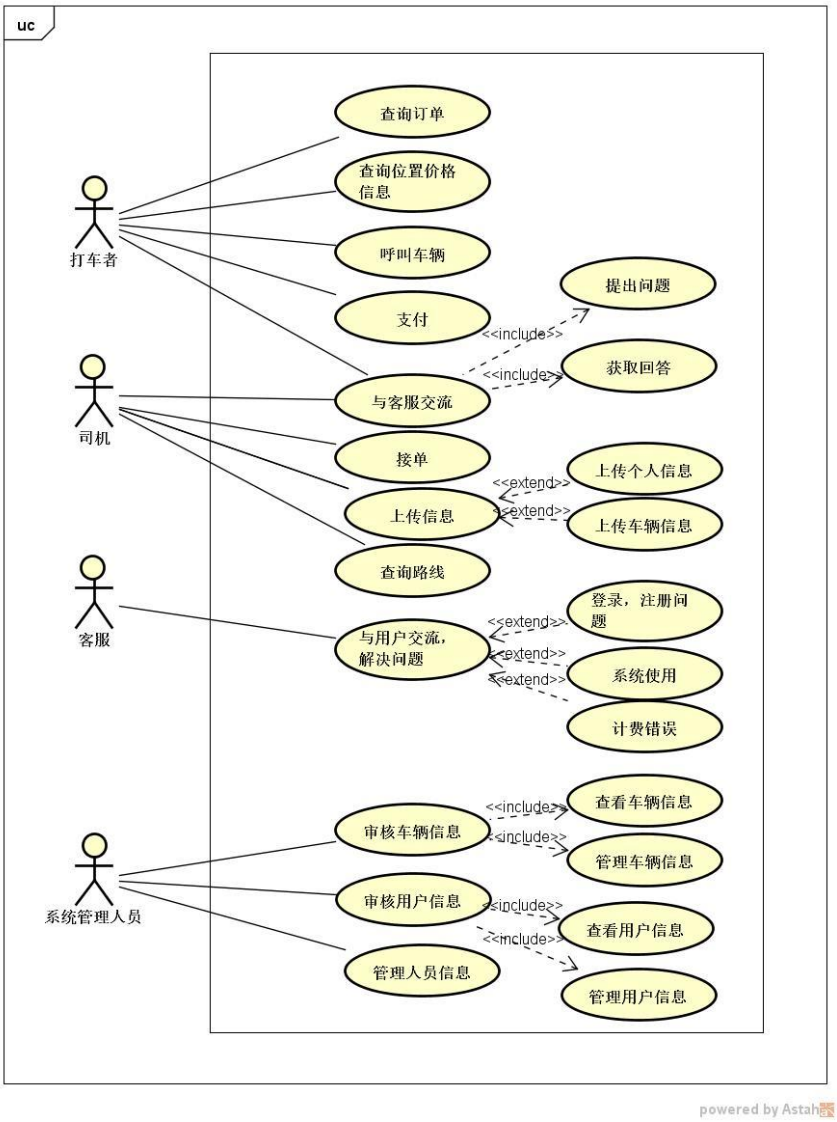
设计要素		主要约束
运行环境软件	操作系统	Windows 7/Linux 7.0 及以上
	数据库	MySQL 14.0 及以上
	Web 服务器	WebLogic
用户端 PC 软件	操作系统	Windows/Linux
	浏览器	Chrome/Firefox/Opera/Safari
用户端手机 app	操作系统	Ios/安卓
开发环境支持	操作系统	Linux Ubuntu 14.10
	开发工具	Myeclipse
	Web 服务器	Weblogic
	CPU	2.4 GHz Intel Core i7
	内存	4GB

4. 用例视图

对软件体系结构的用例视图的描述。 用例视图对于选择作为迭代焦点的场景和/或用例集非常重要。 它描述了代表一些重要的核心功能的场景和/或用例集。 它还描述了一组场景和/或用例，这些场景和/或用例具有实质性的体系结构覆盖范围（行使许多体系结构元素），或者着重或说明了体系结构的特定细节点。用例由打车者，司机，客服，系统管理人员发起。 此外，与外部参与者也有互动： 支付，导航和智能计算当前时间当前地点到目的地的估计价格。

4.1 具有架构重要性的用例

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、子系统以及各项功能的使用和行为。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，设计体系结构用例进行了总结和设计。本打车软件系统的用户角色包括司机、系统管理人员，打车者，客服等不同的用户对应着不同的功能，针对这些用户角色，可以分别画出用例图：



图表名称：具有重要结构意义的用例

打车者：

4.1.1 查询订单

简要说明：该用例允许司机和打车者在登录系统之后查询自己的历史订单信息。

4.1.2 查询位置价格信息

简要说明：该用例允许打车者在打车之前可以先查询从当前地点到目标地点的估计价格，决定是否打车。

4.1.3 呼叫车辆

简要说明：该用例允许打车者如果有打车的需要，可以使用此软件按照规定的流程进行呼叫车辆。在呼叫车辆之前可以首先查看到目的地的价格信息，决定呼叫车辆的类型和是否呼叫车辆。

4.1.4 支付

简要说明：该用例允许在打车的过程结束之后，打车者可以使用本软件对打车的金额进行支付。本打车软件的支付系统与第三方支付系统相连接，支付完毕之后可以生成订单信息。

4.1.5 与客服交流

简要说明：该用例允许打车者，司机在使用本软件的过程中，如果遇到问题，可以在软件与客服进行交流，向客服提出自己遇到的问题，并获得客服的解答。

司机：

4.1.6 接单

简要说明：该用例允许司机在登录此软件后，可以进行在线接单。在接单的过程中，系统首先为司机展示附近的打车请求，司机在浏览打车请求之后，根据自身的情况选择接受订单，并按照系统的指示前往目的地接乘客。
司机可以接单的前提条件是已经提交个人信息和车辆信息并且已经通过审核，成为有资质的司机。

4.1.7 上传信息

简要说明：该用例允许司机执行上传信息的操作。分为上传个人信息和车辆信息。管理员会根据上传的信息，审核司机的资质。只有在上传的信息被审核通过以后，

司机才可以接单并提供驾车服务。

4.1.8 查询路线

简要说明：该用例允许在司机接单并开始提供驾车服务时，本系统为司机规划最优的路线。司机可以查询多条不同的路线而不用退出此系统，方便了驾车。

客服：

4.1.9 与客户交流，解决问题

简要说明：该用例允许客服执行与司机或打车者交流，解决问题的操作。解决的问题包括登录注册问题，系统使用，计费错误等。交流的过程是首先由司机或打车者提出问题，客服告知其解决方案。

系统管理人员：

4.1.10 审核车辆信息

简要说明：该用例允许系统管理人员审核车辆的信息，并确定司机和车辆是否有提供驾车服务的资质。

4.1.11 审核用户信息

简要说明：该用例允许系统管理人员可以审核用户信息。包括审核司机，打车者，客服的用户信息。并确定信息是否合法。

4.1.12 管理人员信息

简要说明：该用例允许系统管理人员可以对人员信息进行管理。

5. 逻辑视图

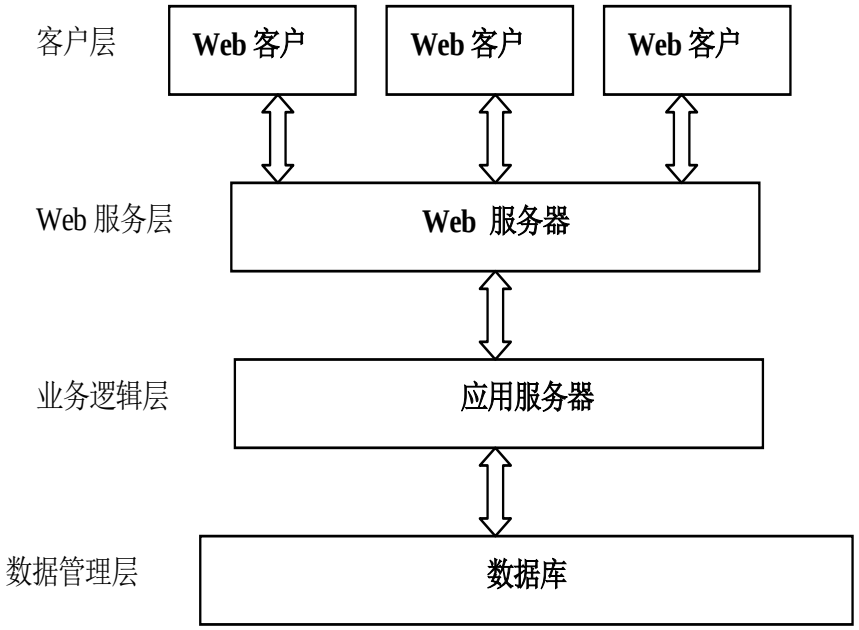
对体系结构逻辑视图的描述。描述最重要的类，它们在服务包和子系统中的组织以及这些子系统在层中的组织。还描述了最重要的用例实现，例如，体系结构的动态方面。可以包括类图以说明在结构上重要的类，子系统，包和层之间的关系。

BTU 打车软件系统的逻辑视图由体系结构概述，模块级体系结构，概念体系结构图，软件包和子系统分层，逻辑数据库需求。

5.1 体系结构概述

1)了满足 BTU 打车软件系统的多种需求，在本项目中将采用四种设计模式进行开发设计。司机和打车者是我们系统的主要使用者，考虑到他们的使用场景，C/S 架构是最佳的选择。
2) 我们的系统要求为系统管理人员提供方便快捷审核司机信息和车辆信息，管理人员信息的功能，而且系统管理员的工作场景固定，是在 pc 机上完成自己的工作。客服和系统管理人员的工作场景一样，也是在 PC 机上进行，所以多层 B/S 架构便成为了最佳的选择。
3) 我们的系统要求具有极高的时效性。每一次数据更新必须快速返回给系统以及用户，而且多个系统部件需要共享数据，选择主动式仓库模式进行架构设计同样是系统需要。
4）为了使得代码结构清晰，层次分明，展示层与应用层、数据层能够清晰有效地进行高效率的交互，系统还应当使用 MVC 模式进行开发。下面对于这几种模式结合系统进行简要的介绍，解释为什么本系统要采用这些模式进行设计开发。

5.1.1 分层的 C/S 模式



客户端-服务器(C/S---Client-Server)模式是满足这种要求的一种结构。分层的 C/S 模式的优点在于可使用性强，故障处理快速，在客户端与服务器的通信失败的情况下可以在用户不知晓的情况下实现恢复。可修改性强，因为每个层面都有其内部逻辑，修改不会影响其它层。

在本打车软件系统的手机端，由于在上下班高峰期使用本软件的用户可能激增，用户的复杂且数量大的操作可能会造成系统出现故障，此时需要快速恢复。因此选择 C/S 模式为本软件的手机 APP 端的开发结构。

5.1.2 三层 B/S 模式

B/S 结构是 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。三层 B/S 结构具有界面统一，使用简单，易于维护，可伸缩性好，分布计算的基础结构可以更充分的利用系统资源等优点，适合用于 BTU 打车软件的 PC 端的开发。

三层结构是伴随着中间件技术的成熟而兴起的，核心概念是利用中间件将应用分为表示层、业务逻辑层和数据存储层三个不同的处理层次，三个层次从逻辑上分，具体的物理分法可以有多种组合。

图 5-1-1-1-2 是三层 B/S 结构的示意图。

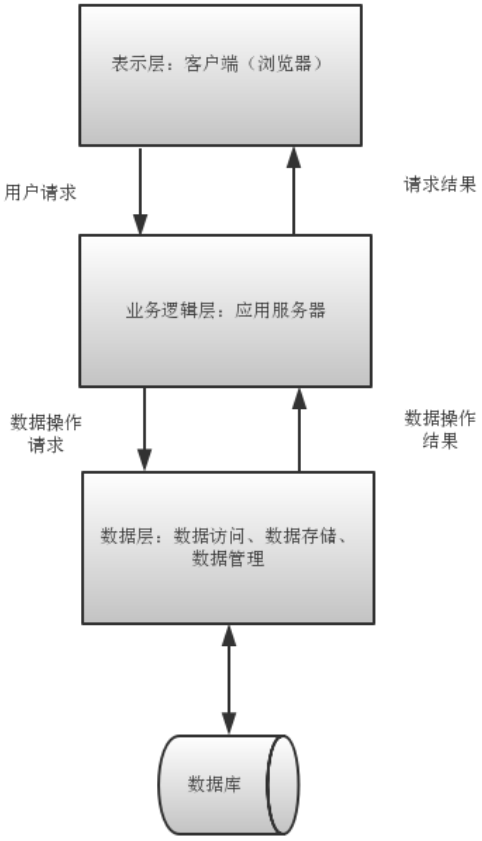


图 5-1-1-1-2 三层 B/S 体系结构示意图

5.1.3 MVC 模式

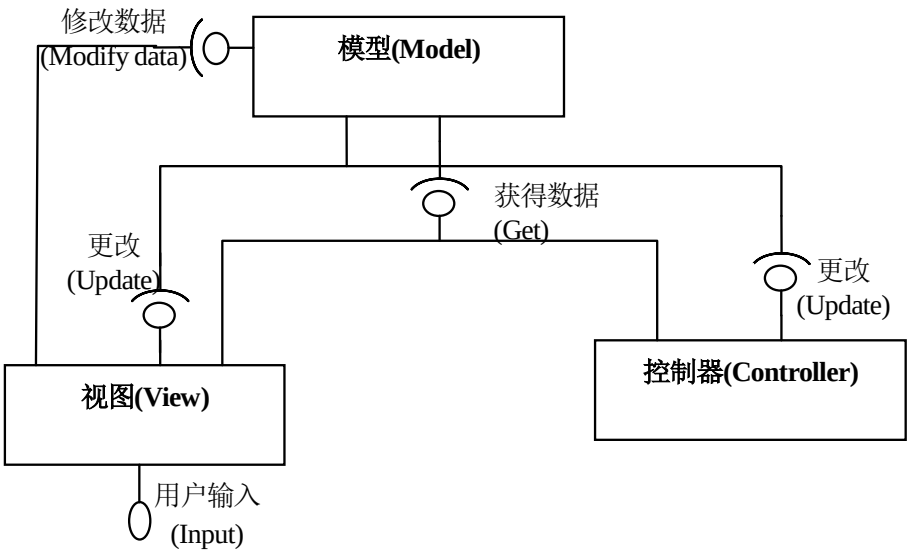
BTU 打车软件系统需要提供多个用户界面(User Interface)，而每个用户界面可能只需要反应一部分应用数据。当这些数据改变时，要能够自动和灵活地反应到不同的用户界面上。这就需要能够很容易地修改其中的一些用户界面，而不需要修改与数据相关联的应用逻辑。

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。



5.1.4 主动式仓库模式

主动式仓库模式是共享仓库模式的一个变体，要求把仓库中发生的特殊事件主动地通知给客户端。主动仓库要维护客户端的注册表，并通过适当的提示机制提示客户端。
主动式仓库模式具有数据共享，以及两个不相邻部件之间的信息交流方便的优点。在我们的系统中，数据经常是被动的访问，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，客户端必须立即得到通知。所以，在 BTU 打车软件系统中，使用主动式仓库模式进行数据的存储与访问。

5.2 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 5-2 所示。

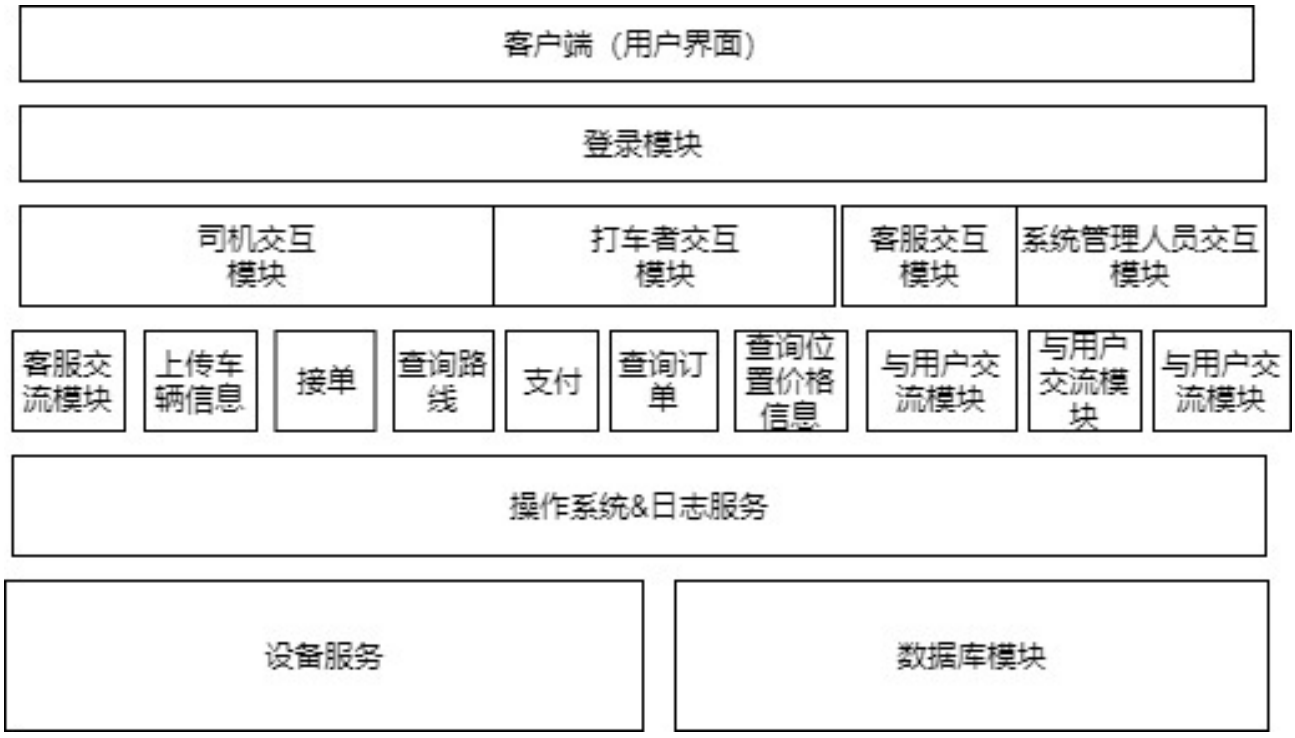


图 5-2 系统级模块级体系结构示意图

5.3 BTU 打车软件的概念级体系结构图

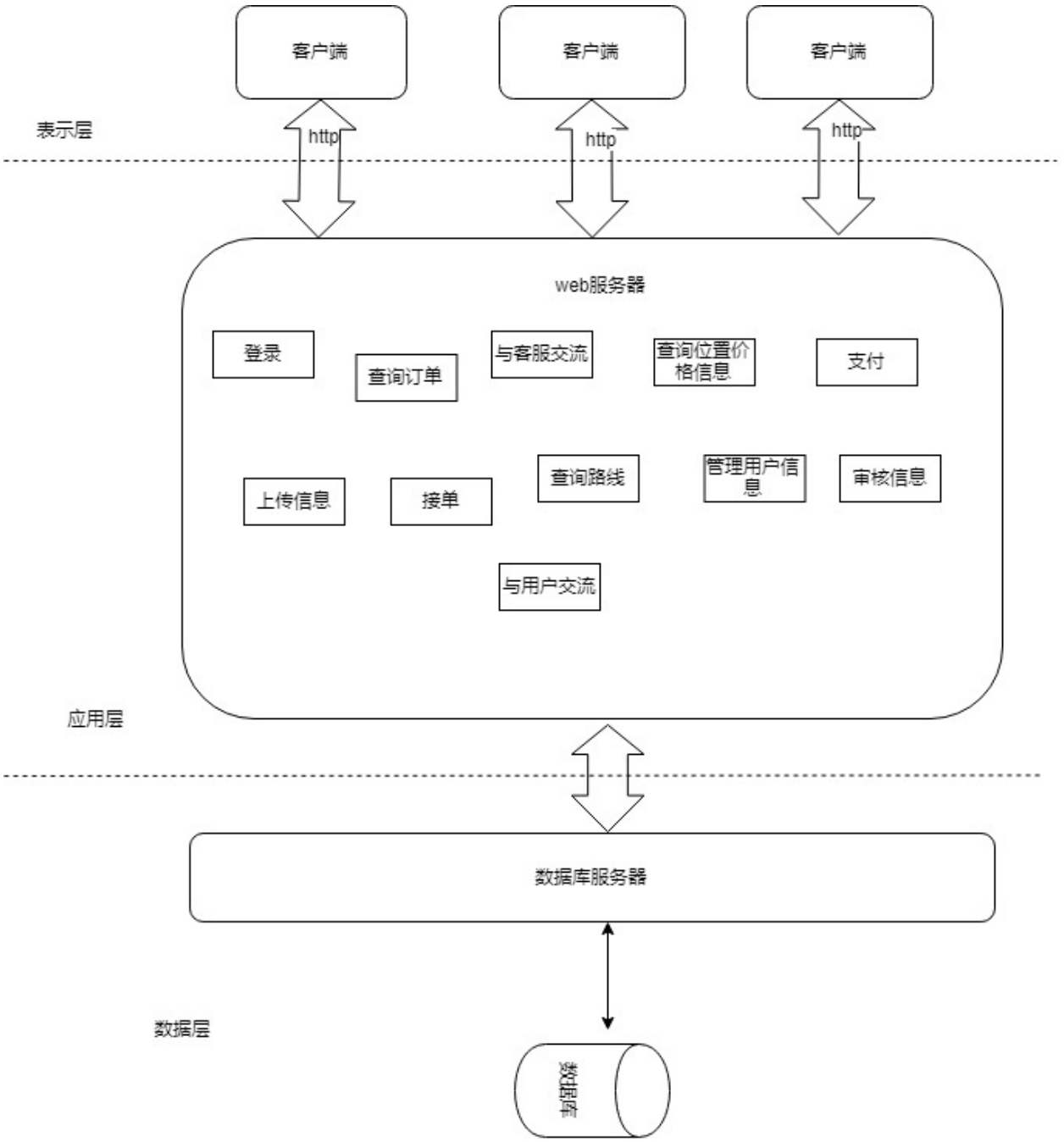


图 5-3 概念级体系结构图

5.4 软件包和子系统分层

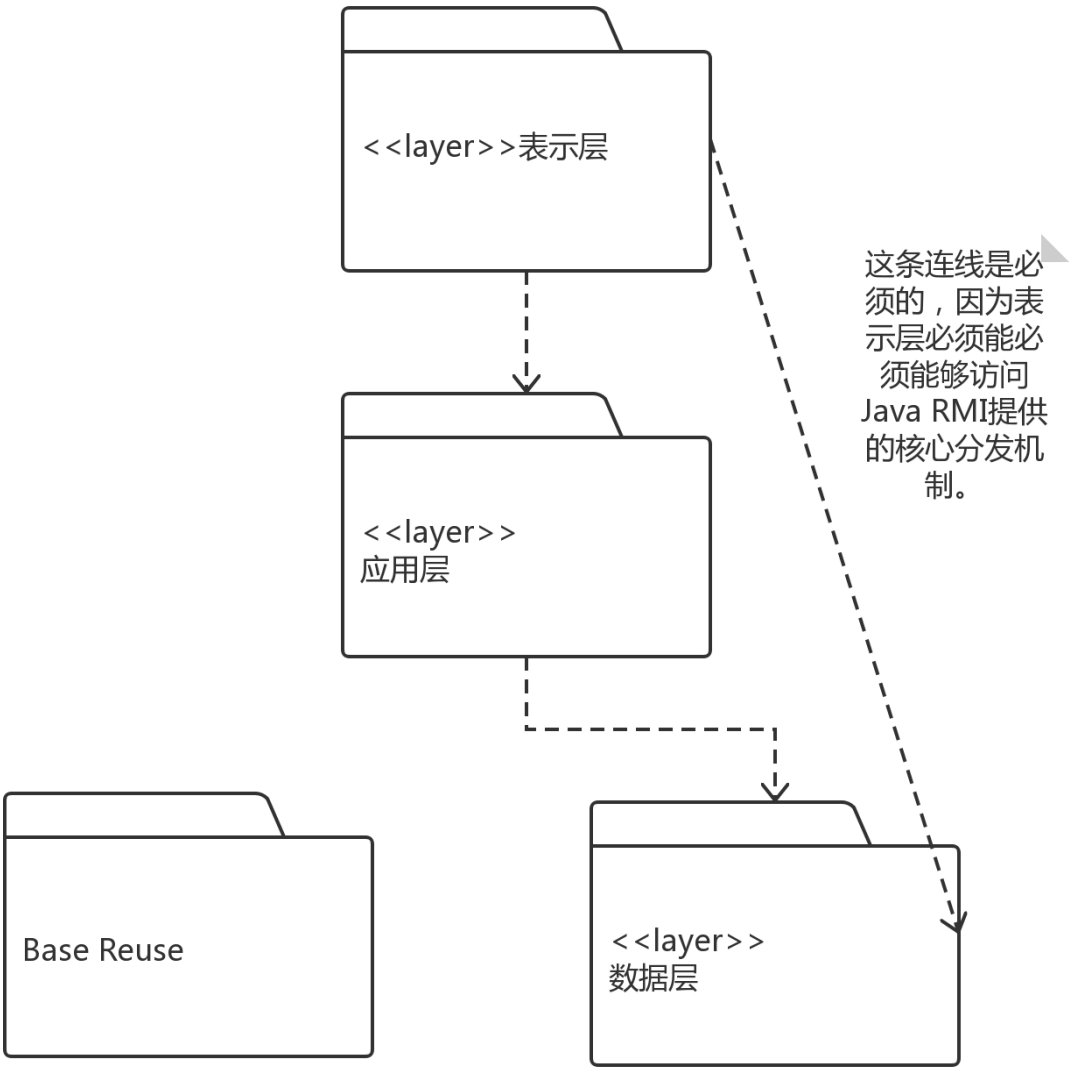


图 5-4 **软件包和子系统**分层

对 5-3 和 5-4 图的说明:

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。

5.5 逻辑数据库需求

1.基本需求

系统推荐使用 mysql。费用较低，用户数量大，可靠性高。

若经费预算允许，可以使用大型数据库，可靠性高，稳定性好。

数据表的字段类型的定义应该具有一定的灵活性，保证可修改性。

满足上述前提的情况下，数据库存储应该尽量节省空间。

2.数据库需求

数据库需求设计分为两部分：概念结构设计和逻辑结构设计。概念结构设计指的是画出 E-R 模型。将概念结构进一步转化为某一 DBMS 所支持的数据模型，然后根据逻辑设计的准则、数据的语义约束、规范化理论等对数据模型进行适当的调整和优化，形成合理的全局逻辑结构，并设计出用户子模式。

E-R 图：

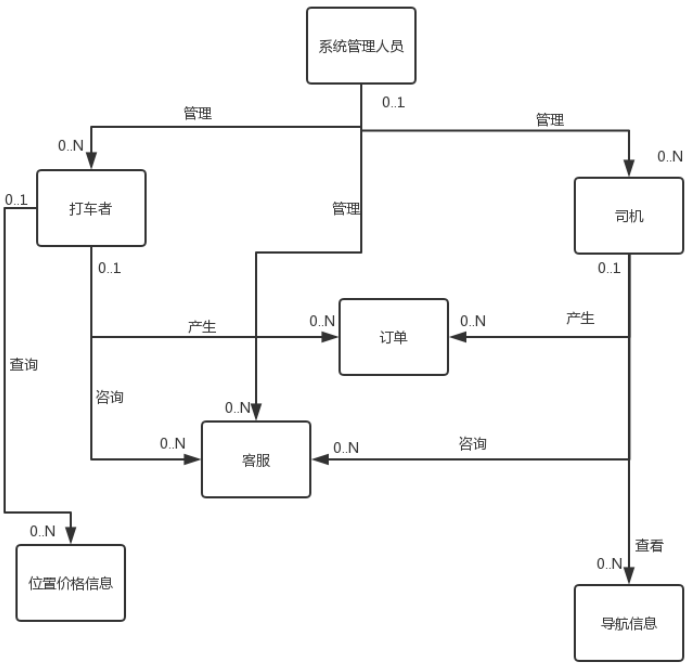


图 5-5 数据库 E-R 图

6. 进程视图

对体系结构的过程视图的描述。 描述系统执行中涉及的任务（进程和线程），它们的交互作用和配置。 还描述了对象和类对任务的分配。
本文中的过程模型按照用户的类型进行分类，每种不同的用户类型都有一个不同的过程模型，表示了系统和用户进行交互的过程。
过程模型说明了组织为可执行过程的打车者应用类，司机应用类，系统管理人员应用类，客服应用类。 存在打车者交互，司机交互，系统管理人员交互以及客服交互的过程。

6.1 Processes

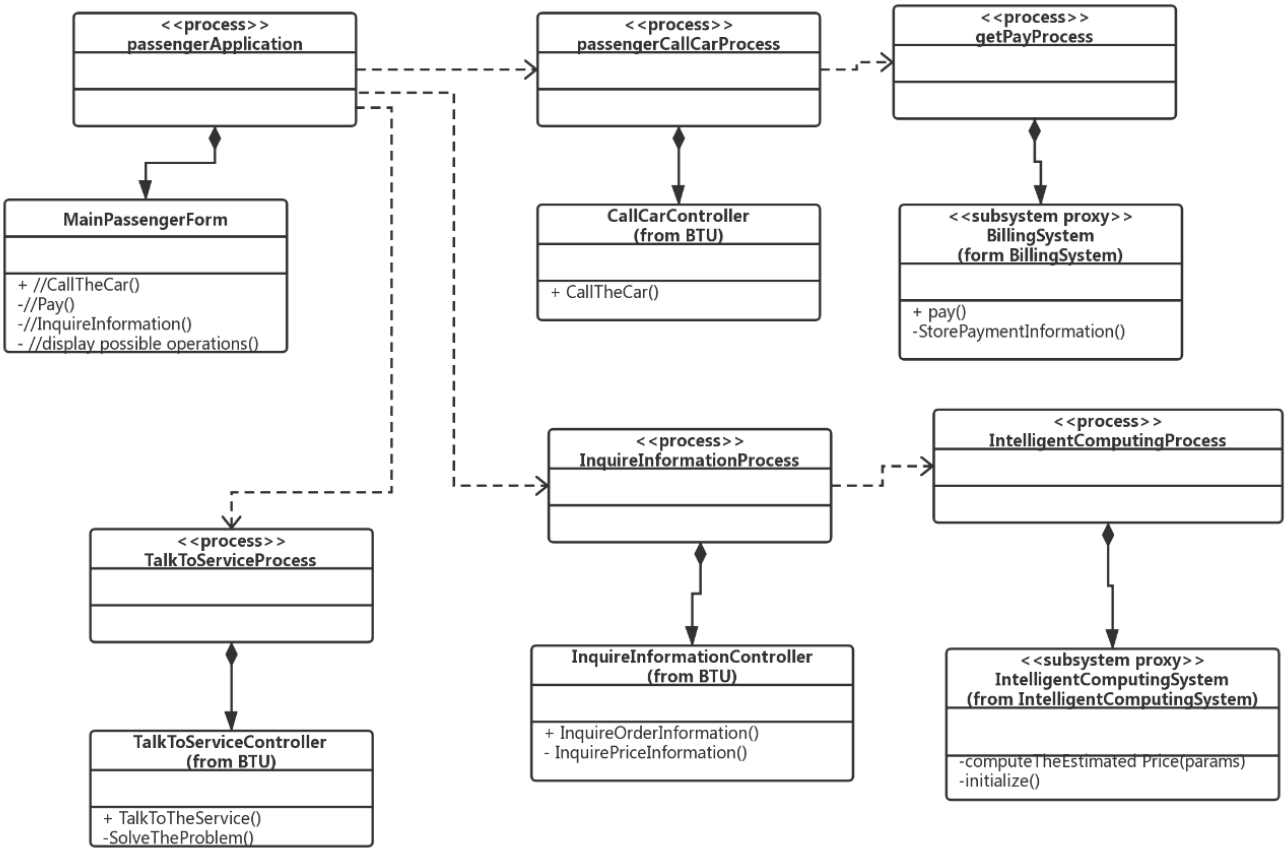


图 6-1-1 打车者进程视图

打车者进程视图：

6.1.1passengerApplication

管理打车者交互的功能，包括用户界面处理和与业务流程的协调。对于当前正在使用系统的每个打车者，都有一个此过程的实例。

6.1.2MainPassengerForm

控制打车者应用程序的界面。控制打车者使用的表格系列。

6.1.3passengerCallCarProcess

对于当前正在呼叫车辆的每个打车者，都有一个此过程的实例。这个过程支持了叫车的用例，打车者通过这个过程发出打车的请求。

6.1.4CallCarController

这支持叫车用例，该用例允许打车者在当前时间，当前位置进行叫车。如果打车者不确定是否打车，可以在当前界面查看当前位置到目的地的估计价格，决定是否打车以及选择打车的类型。

6.1.5getPayProcess

对于每个打车完成在进行支付的用户都有一个这个流程的实例。这个流程允许用户通过此软件进行支付打车费用。
完成该流程需要软件实现第三方的支付接口。

6.1.6BillingSystem

支付系统支持为打车者在本次打车过程计费，支付，并提交账单到数据库保存。

6.1.7IntelligentComputingProcess

智能计算流程。在打车者打车之前，可以通过本打车软件查看到目的地的估计价格。每次查看价格，都有一个此过程的实例。

6.1.8IntelligentComputingSystem

智能计算系统。本软件与第三方系统-智能计算系统相连接。在用户查询到目的地的估计费用的时候，本软件可以调用该系统计算结果，并返回给用户的查询界面。

6.1.9InquireInformationProcess

每当用户需要通过该软件获取信息的时候，会实例化一个此流程。

6.1.10InquireInformationController

查询信息的控制器。每当用户查询信息的时候，就会调用此流程，完成查询信息的过程。
用户包括打车者，司机，客服。查询的信息包括用户的个人信息，订单信息等。

6.1.11TalkToServiceProcess

每当用户（打车者，司机）和客服交流解决问题的时候，会实例化一个此流程。

6.1.12TalkToServiceController

对应了与客服交流的用例。当用户有问题需要解决的时候，系统调用TalkToServiceController，实现功能。

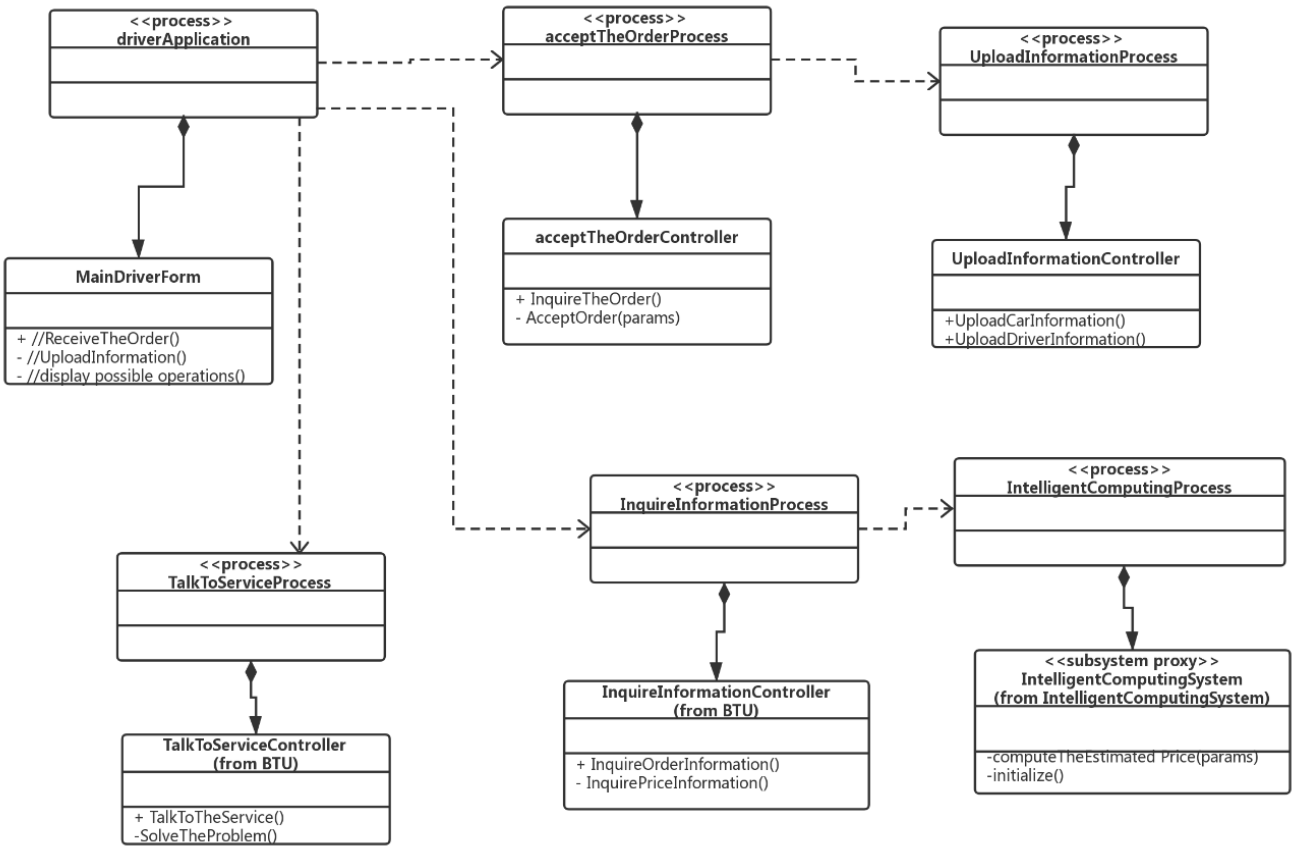


图 6-1-2 司机进程视图

司机进程视图：

6.1.13driverApplication

管理司机交互的功能，包括用户界面处理和与业务流程的协调。对于当前正在使用系统的每个司机，都有一个此过程的实例。

6.1.14MainDriverForm

控制司机应用程序的界面。控制司机使用的表格系列。

6.1.15acceptTheOrderProcess

接单流程。每个司机在进行接单的操作的时候，都会实例化一个此流程。

6.1.16acceptTheOrderController

接受订单的控制器。里面包含了查询附近订单，接受订单，根据订单确定接下来的操作等。此控制器在司机接单操作的时候被调用，实现了系统的接单的功能。

6.1.17UploadInformationProcess

上传个人信息流程。每当司机在本打车软件中上传个人信息和车辆信息时，会实例化一个此流程。

6.1.18UploadInformationController

上传信息控制器。这支持上传信息用例。该用例允许司机上传个人信息和车辆信息接受管理员的审核。审核通过之后才具有司机的资质。

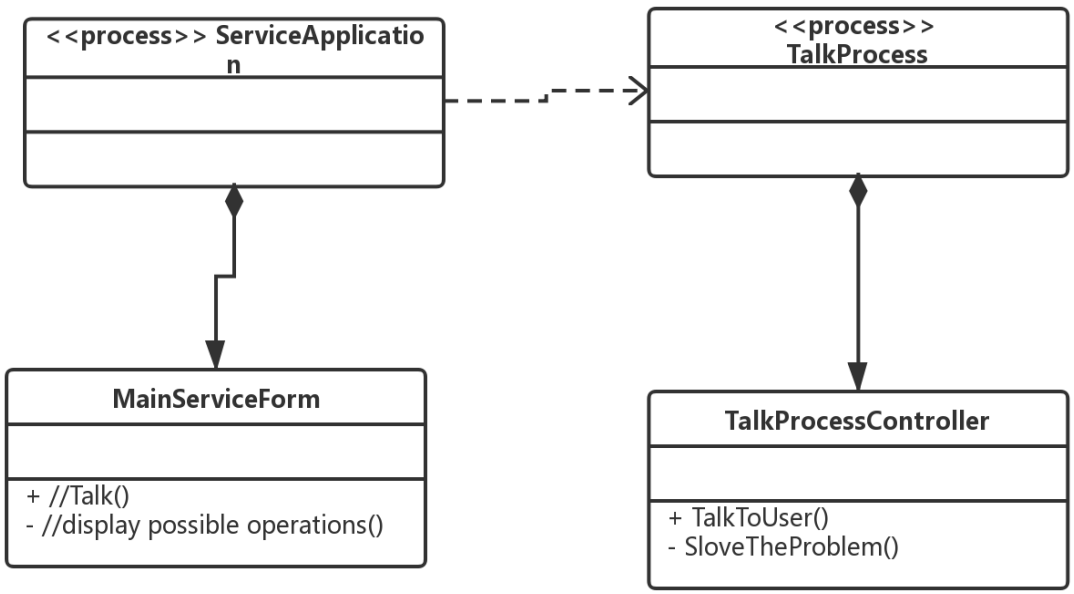


图 6-1-3 客服进程视图

客服进程视图：

6.1.19ServiceApplication

管理客服交互的功能，包括用户界面处理和与业务流程的协调。对于当前正在使用系统的每个客服，都有一个此过程的实例。

6.1.20MainServiceForm

控制客服应用程序的界面。控制客服使用的表格系列。

6.1.21TalkProcess

交流流程。每当一个客服通过此软件和司机或打车者交流的时候，会实例化一个此流程。此流程包括用户提出问题，客服根据问题提出解决方案，在有需要的时候记录下该问题。

6.1.22TalkProcessController

交流控制器。对应了与用户交流的用例。该用例允许客服与用户交流解决他们的问题。问题主要为登录，注册问题，计费错误问题，使用问题等。

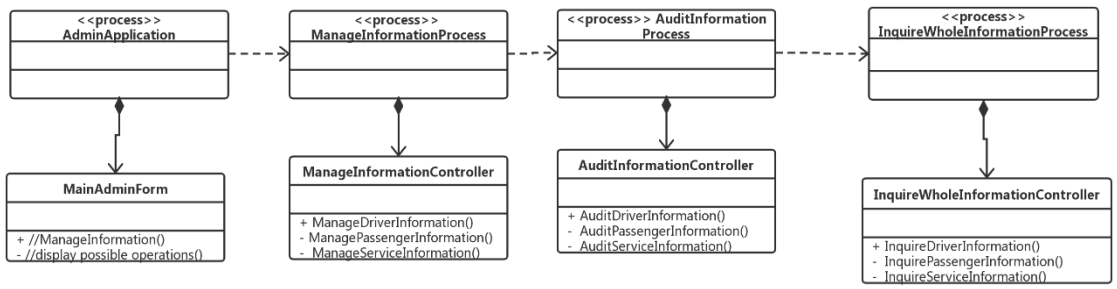


图 6-1-4 系统管理人员进程视图

管理员进程视图：

6.1.23AdminApplication

管理系统管理人员交互的功能，包括用户界面处理和与业务流程的协调。对于当前正在使用系统的每个系统管理人员，都有一个此过程的实例。

6.1.24MainAdminForm

控制系统管理人员应用程序的界面。控制系统管理人员使用的表格系列。

6.1.25ManageInformationProcess

管理信息流程。 每当管理员登录，在此软件进行信息的管理的时候，会实例化此流程。

6.1.26ManageInformationController

管理信息控制器。该控制器对应了管理信息的用例。管理员管理信息包括管理人员信息，管理车辆信息，管理司机信息等。其中，管理员在审核信息之后可以对信息进行管理，比如确定注册本软件的司机是否有司机资质等。

6.1.27AuditInformationProcess

审核信息流程。每当管理员登录，在此软件进行信息的审核的时候，会实例化此流程。

6.1.28AuditInformationController

审核信息控制器。该控制器实现了审核信息的用例。此用例允许管理员进行信息的审核。审核的信息包括司机个人信息，车辆信息等。根据审核的结果确定司机是否有资质进行接单并提供驾车服务等。

6.1.29InquireWholeInformationProcess

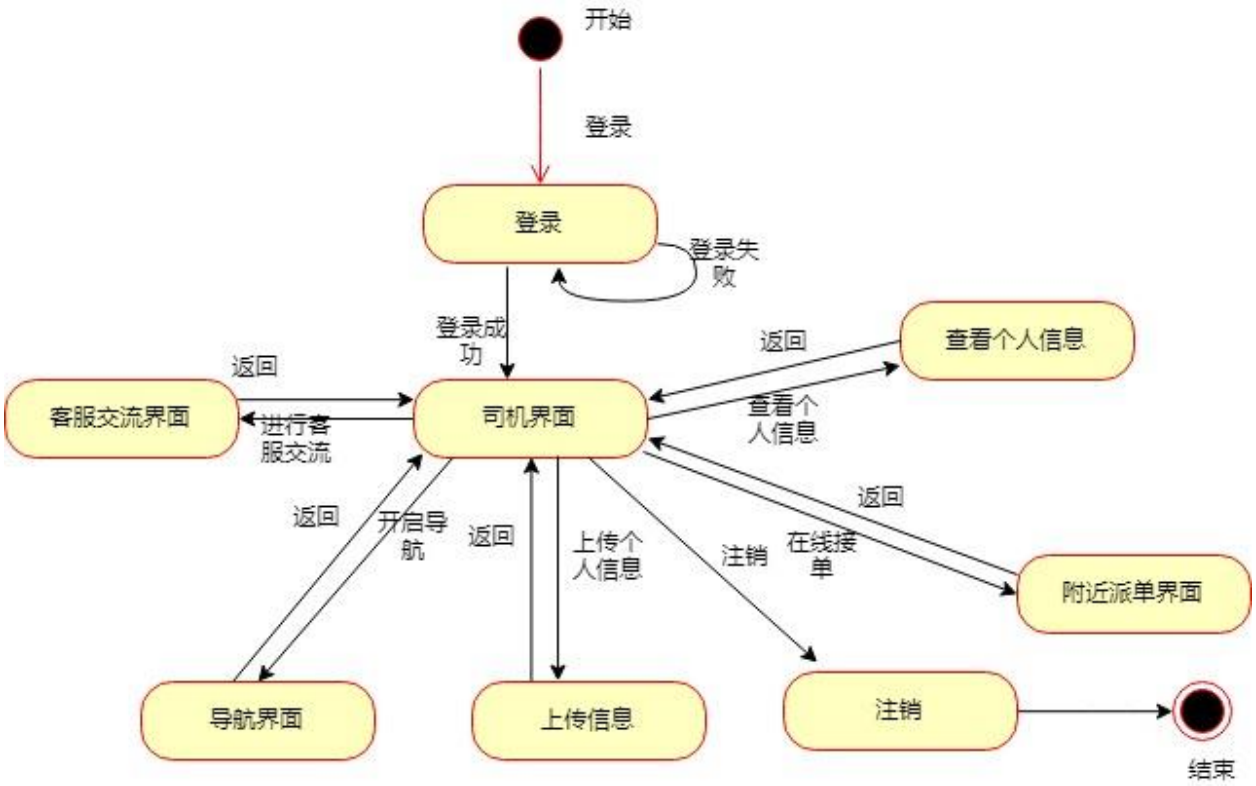
查询所有信息流程。每当管理员登录，在此软件进行信息的查询的时候，会实例化此流程。

6.1.30InquireWholeInformationController

审核信息控制器。该控制器允许管理员查询系统的所有信息。通常伴随着信息的管理和审核一起使用。

6.2 图表名称：设计元素的过程

6.2.1 司机行为视图



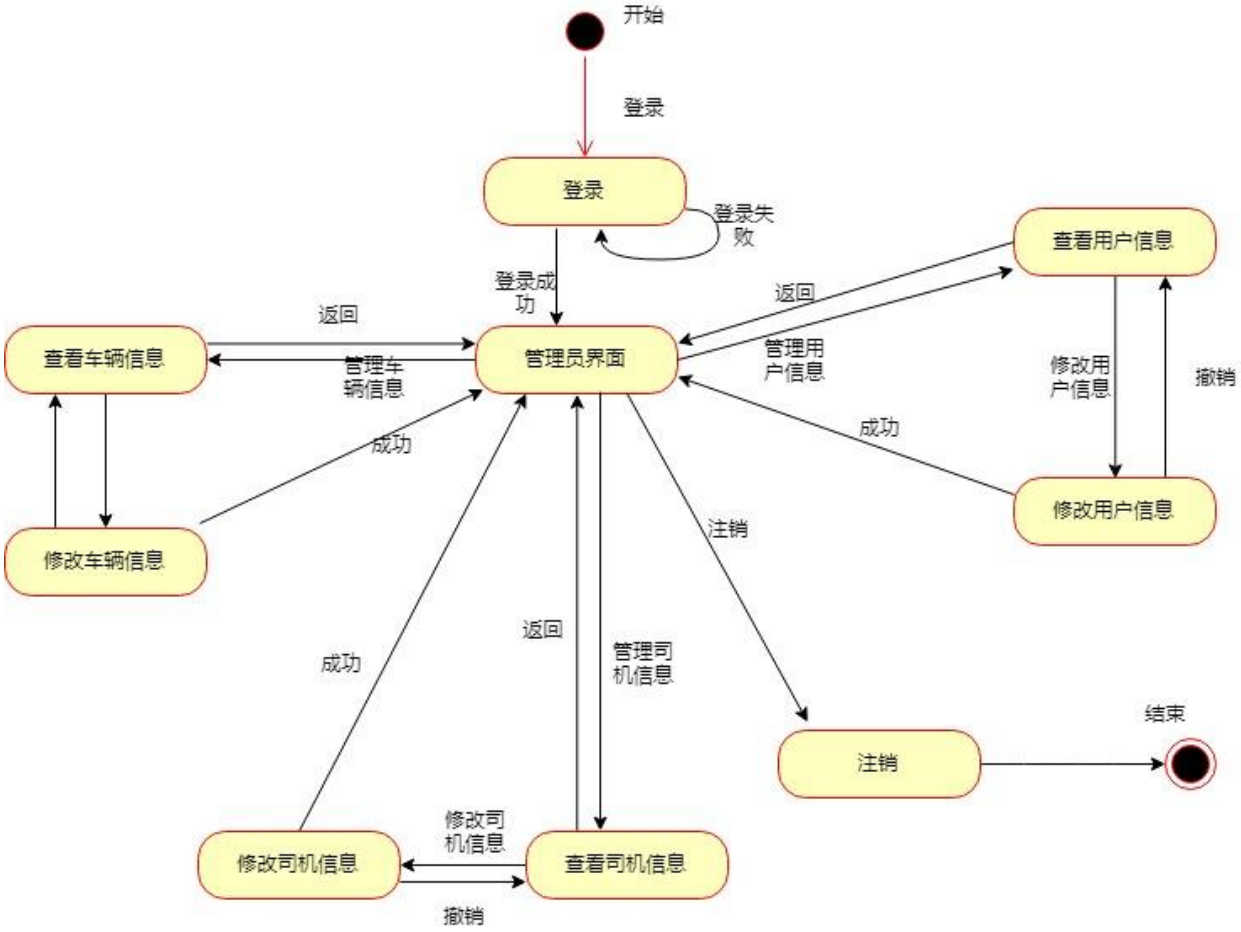
司机在开始状态开启本打车软件，首先进行登录操作，如果登录成功，则进入司机界面，进行下一步操作。如果登录失败，返回登录界面进行下一次登录的操作。登录成功进入司机界面后，司机可以进行与客服交流，查看个人信息，使用导航，上传信息，在线接单等操作。

1）与客服交流，进入客服交流界面，与客服发消息，提出自己的问题并获得解决方案，完毕后返回司机界面。

2）使用导航，首先开启软件中的导航功能，进入导航界面，界面向司机展示路线，实时路况等信息，方便司机顺利到达打车者所在地和目的地，完毕后返回司机界面。

3）上传信息：司机进入上传信息的界面，上传个人信息和车辆信息并等待审核，完毕后返回司机界面。
4）查看个人信息，司机进入个人信息界面，进行查看，查看完毕后返回司机界面。
5）在线接单：司机打开软件中的附近派单信息界面，查看界面展示的打车请求，订单等消息，根据自己的需求进行接单。
6）在司机完成所有操作并停止使用此软件时，注销账号并退出软件。

6.2.2 系统管理人员行为视图



系统管理人员在开始状态开启本打车软件，首先进行登录操作，如果登录成功，则进入系统管理人员界面，进行下一步操作。如果登录失败，返回登录界面进行下一次登录的操作。
登录成功进入系统管理人员界面后，系统管理人员可以进行查看车辆信息，修改车辆信息，查看司机信息，修改司机信息，查看用户信息，修改用户信息等操作，实现用例中审核，管理信息的功能。

1）审核车辆信息

审核车辆信息包括查看车辆信息和修改车辆信息。管理员在软件中进入审核车辆信息界面。首先查看车辆信息，按照规定的标准进行审核。审核车辆信息后，发布审核结果，表示此车辆是否有资质提供服务，在原来的车辆信息的车辆状态处做修改并发布审核结果。完毕后返回系统管理人员界面。

2）审核司机信息

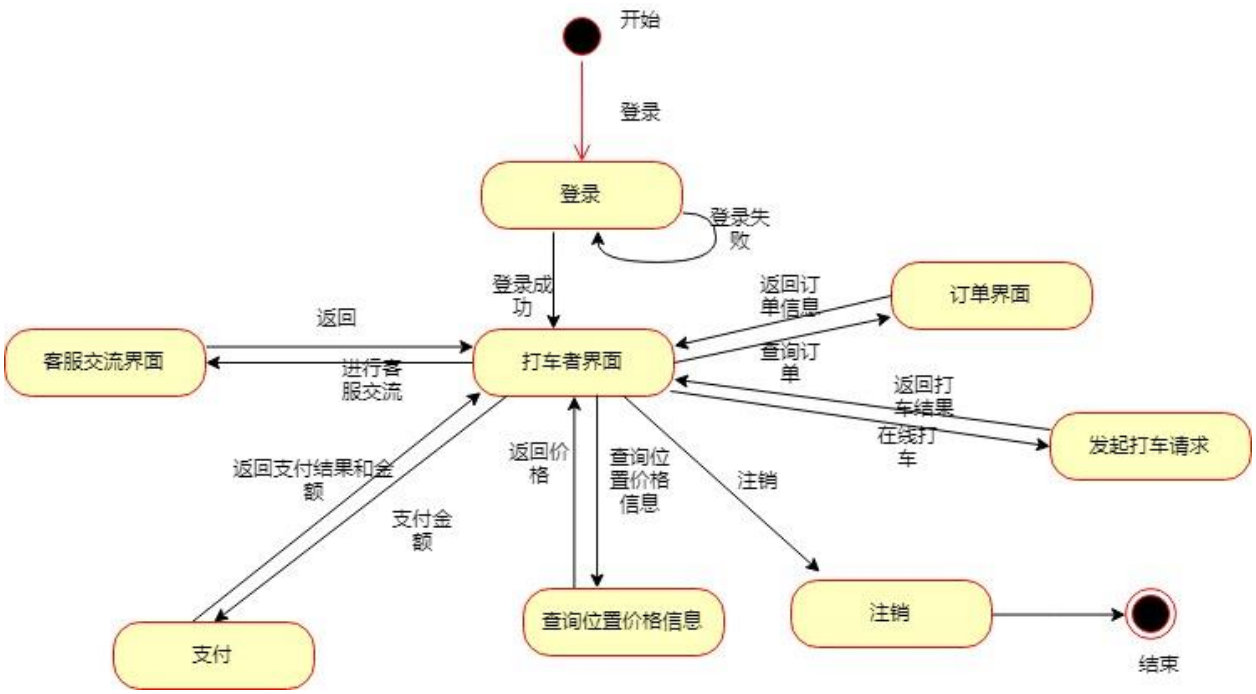
审核司机信息包括查看司机信息和修改司机信息。管理员在软件中进入审核司机信息界面。首先查看司机信息，按照规定的标准进行审核。审核司机信息后，发布审核结果，表示此司机是否有资质提供服务，在原来的司机信息的司机状态处做修改并发布审核结果，完毕后返回系统管理人员界面。

3）管理人员信息

管理员进入管理人员信息的界面，查看用户（打车者，司机）的信息，如果有必要，对其做出修改，并保存修改，完毕后返回系统管理人员界面。

4）在系统管理人员完成所有操作并停止使用此软件时，注销账号并退出软件。

6.2.3 打车者行为视图



打车者在开始状态开启本打车软件，首先进行登录操作，如果登录成功，则进入打车者界面，进行下一步操作。如果登录失败，返回登录界面进行下一次登录的操作。

1）与客服交流，进入客服交流界面，与客服发消息，提出自己的问题并获得解决方案，完毕后返回用户界面。

2）支付，打车者在完成一笔订单之后，进行支付操作。首先进入支付界面，查看金额，并支付金额，支付完毕后系统返回支付结果和金额，完毕后返回打车者界面。

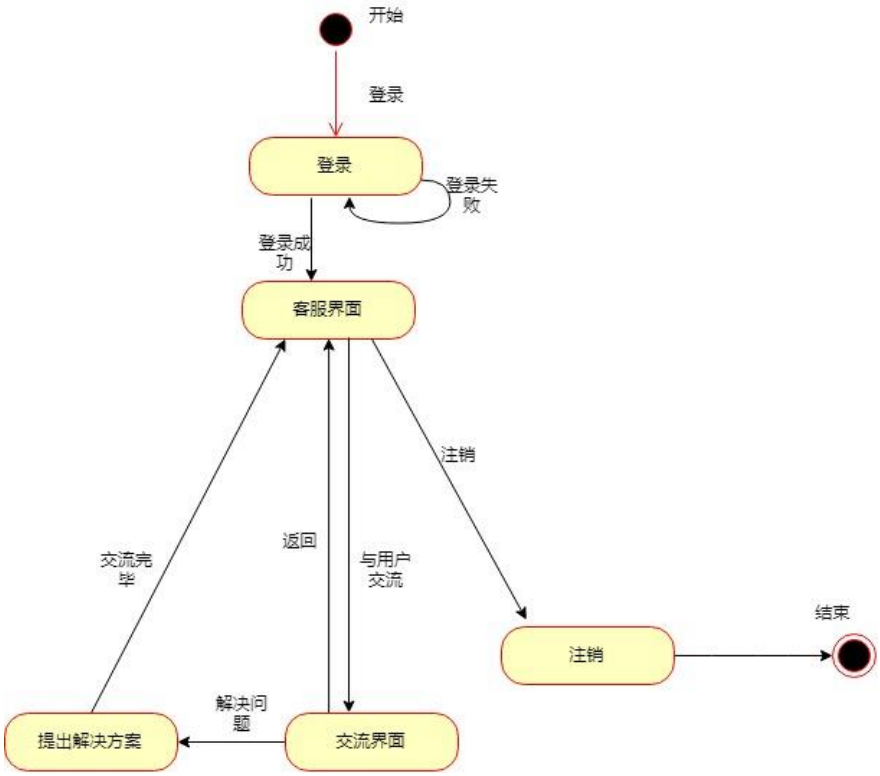
3）查询位置价格信息，打车者在打车之前可以查询到目的地的估计价格作为参考。首先进入查询位置价格信息的页面，输入目的地，系统计算后返回位置价格信息，完毕后返回打车者界面。

4）在线打车，打车者首先进入发起打车请求界面，输入目的地，选择打车类型和车型，确认后打车，当在线打车完毕时，返回打车者界面。

4）查看订单，打车者进入个人订单界面，进行查看，完毕后返回用户界面。

6）在用户完成所有操作并停止使用此软件时，注销账号并退出软件。

6.2.4 客服行为视图



客服在开始状态开启本软件，首先进行登录操作，如果登录成功，则进入客服界面，进行下一步操作。如果登录失败，返回登录界面进行下一次登录的操作。
在登录成功之后，可以执行与用户交流的操作。当客服与用户交流时，首先进入与用户交流的界面，获得用户提出的问题，回复解决方案，交流完毕后返回客服界面。
在客服完成所有操作并停止使用此软件时，注销账号并退出软件。

6.3 过程模型到设计模型的依赖

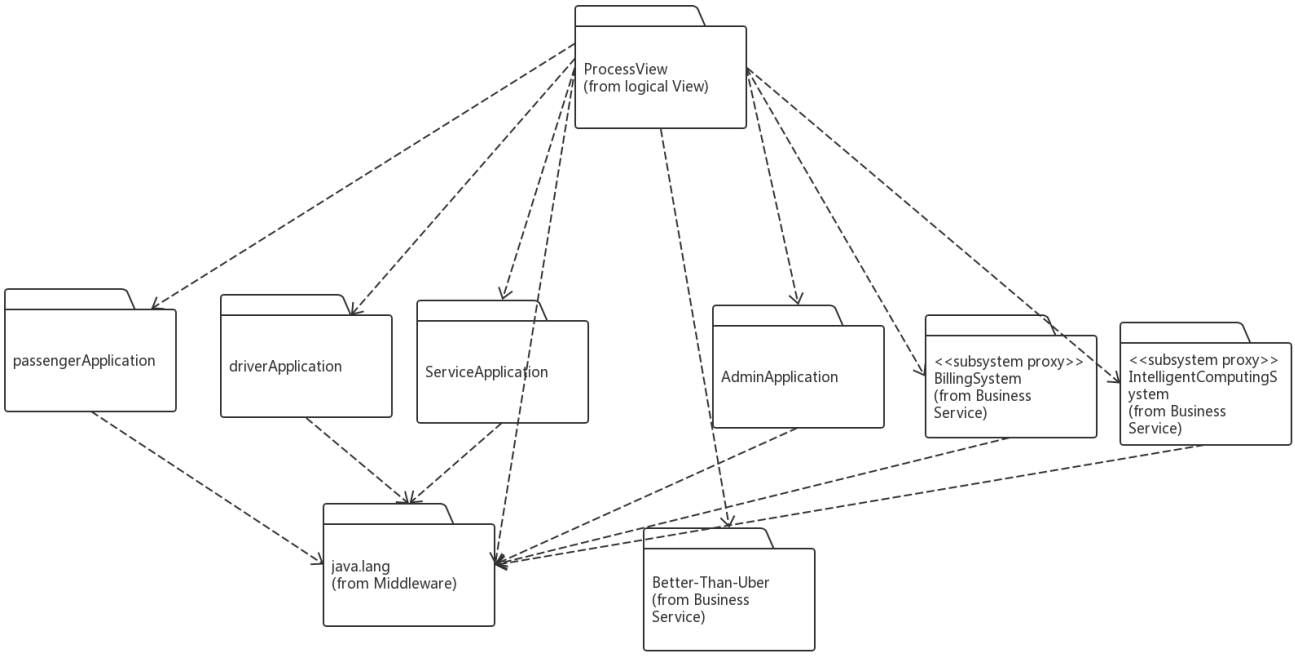


图 6-3：设计模型相关性的过程模型

说明:

1）在进程视图中，按照用户角色的不同，划分为打车者交互，司机交互，系统管理人员交互，客服交互等模块。为了方便本打车软件的支付和计算到目的地的估计价格，系统还与支付系统和智能计算系统相连接。

2）上述模块依赖于来自于中间层的 java.lang 包。

3）逻辑视图依赖于 BTU 打车软件的应用层。

6.4 实施流程

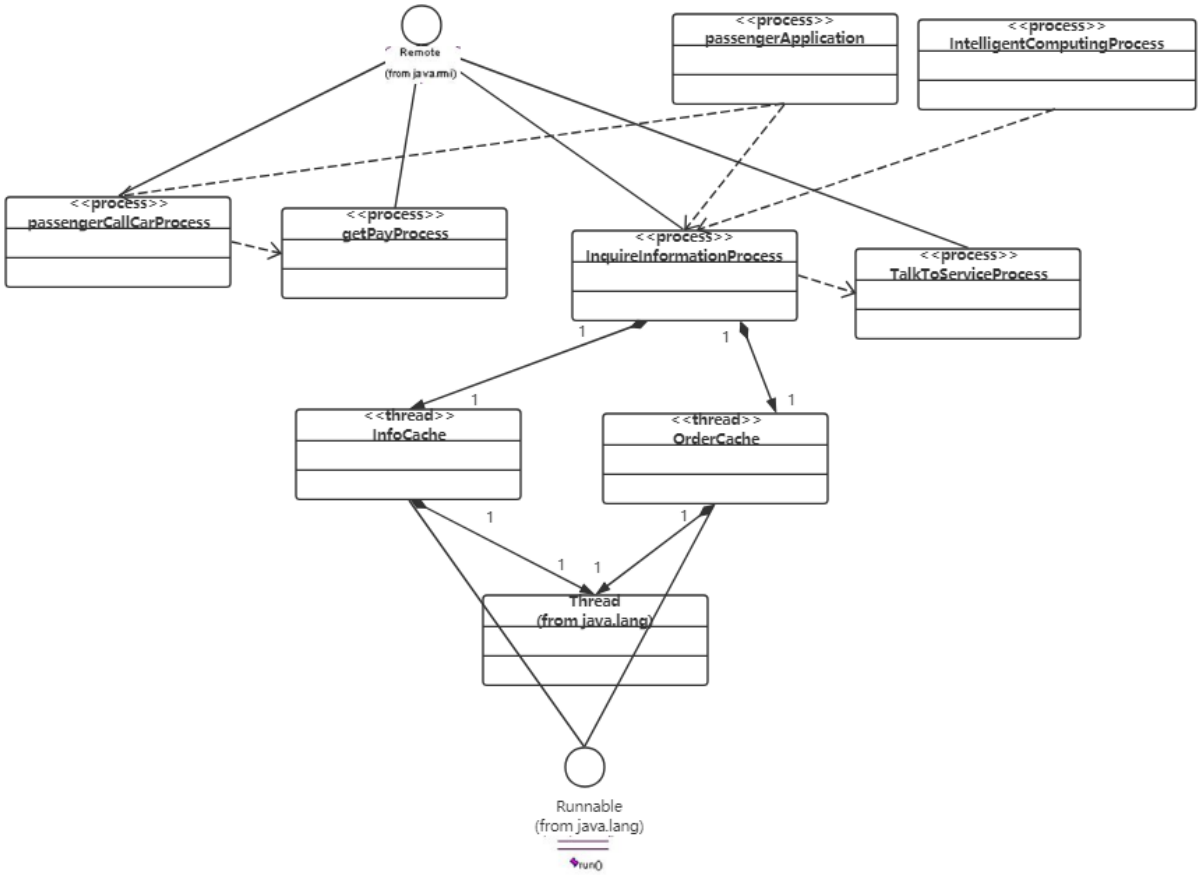


图 6-4-1 打车者实施流程

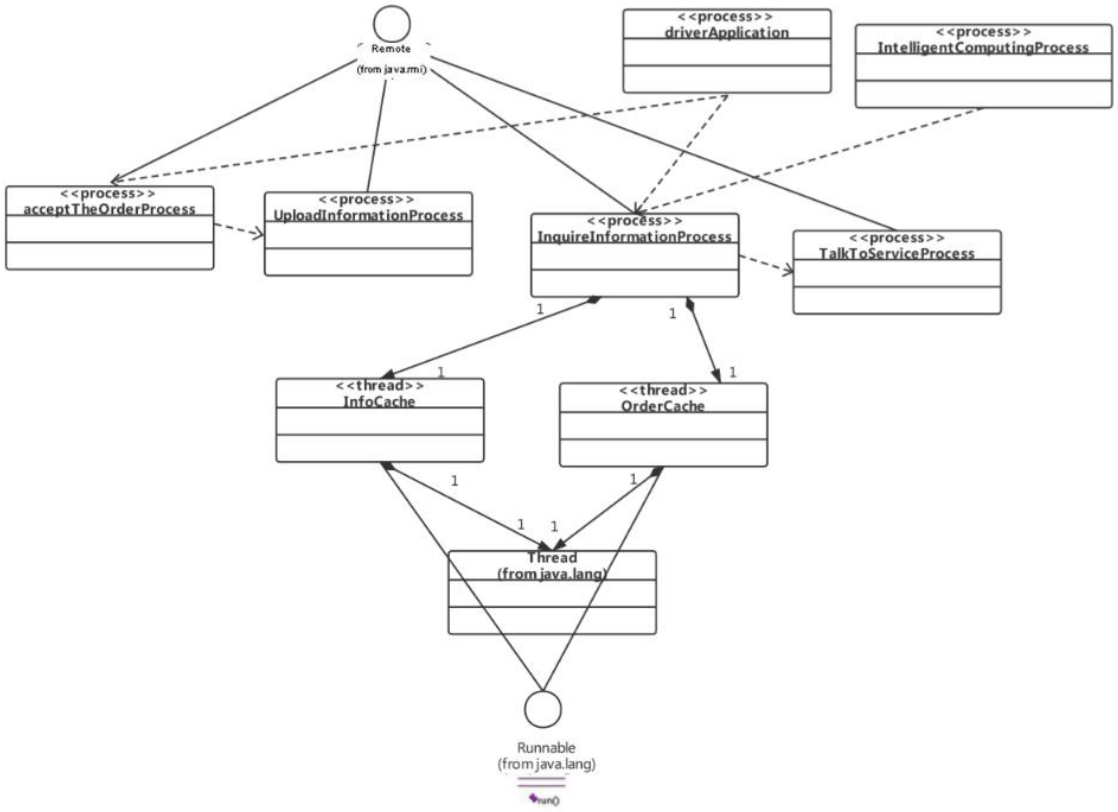

Remote
(from java.rmi)
<<process>>
driverApplication
<<process>>
IntelligentComputingProcess
<<process>>
acceptTheOrderProcess
<<process>>
UploadInformationProcess
<<process>>
InquireInformationProcess
<<process>>
TalkToServiceProcess
1
1
1
1
<<thread>>
InfoCache
<<thread>>
OrderCache
1
1
1
1
Thread
(from java.lang)
Runnable
(from java.lang)
run()

图 6-4-2 司机实施流程

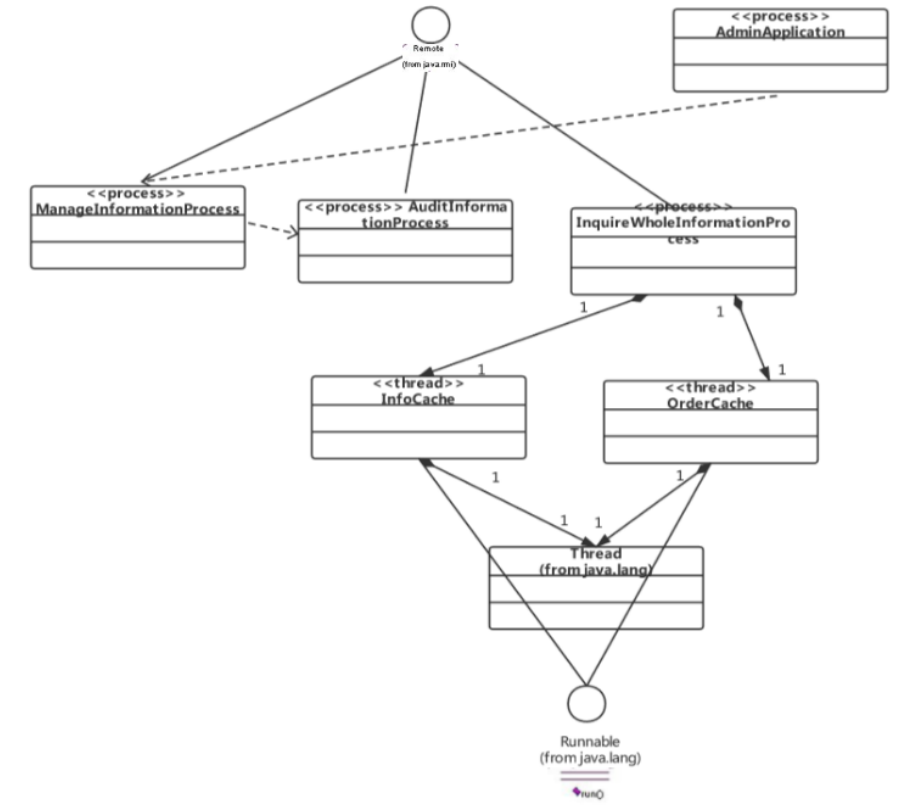


图 6-4-3 系统管理人员实施流程

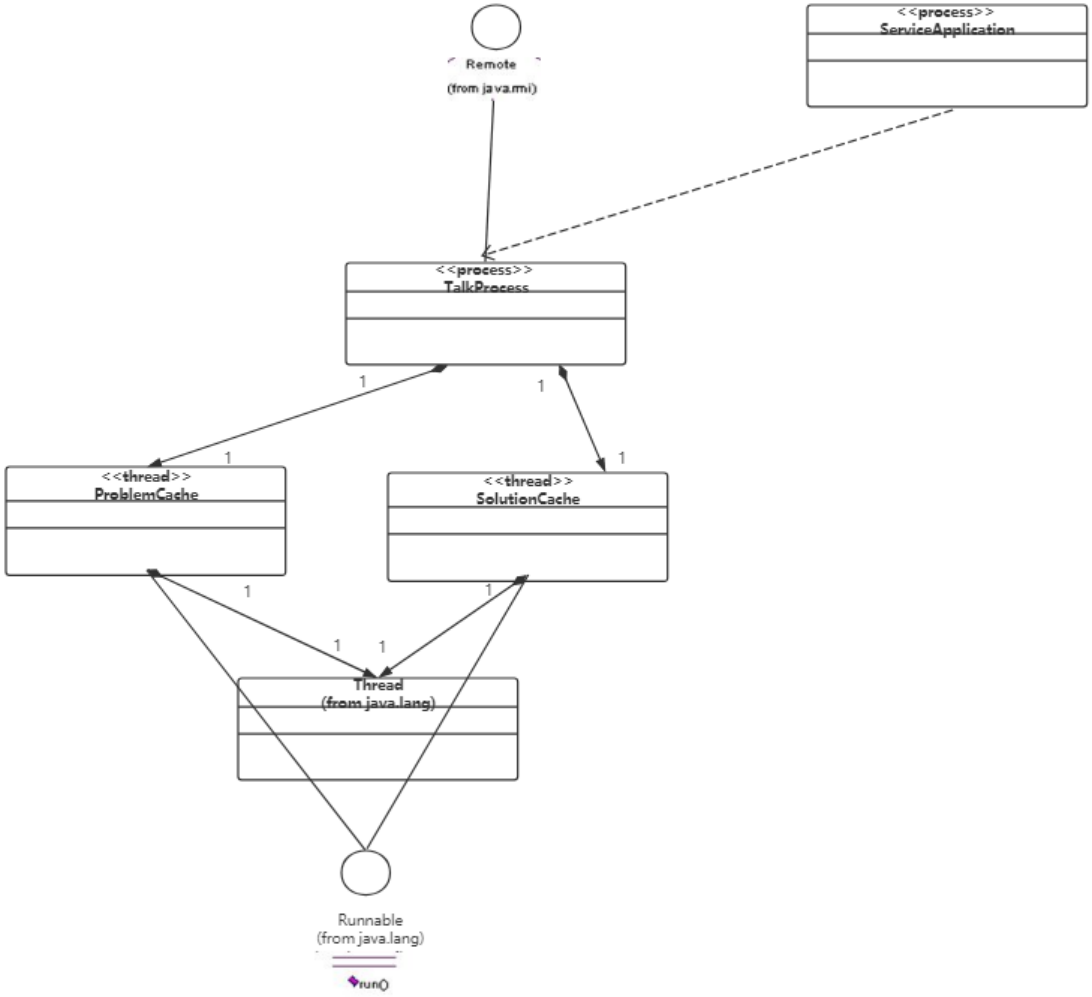


图 6-4-4 客服实施流程

对上述图片的说明：

6.4.1 远程

*远程接口用于识别所有远程对象。任何作为远程对象的对象都必须直接或间接实现此接口。远程接口中仅指定了那些方法。

*实现类可以实现任意数量的远程接口，并且可以扩展其他远程实现类。

6.4.2 可运行

* Runnable 接口应该由实例旨在由线程执行的任何类实现。该类必须定义一个没有参数的方法，称为 run。

*此接口旨在为希望在活动状态下执行代码的对象提供通用协议。例如，Runnable 由 Thread 类实现。

*处于活动状态仅表示线程已启动但尚未停止。

6.4.3 线程

*线程是程序中的执行线程。 Java 虚拟机允许应用程序具有多个并发运行的执行线程。

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

7. 部署视图

体系结构的部署视图的描述描述了最典型的平台配置的各种物理节点。 还描述了任务（从“过程”视图）到物理节点的分配。

本节按物理网络配置进行组织。 部署图说明了每个此类配置，然后将进程映射到每个处理器.

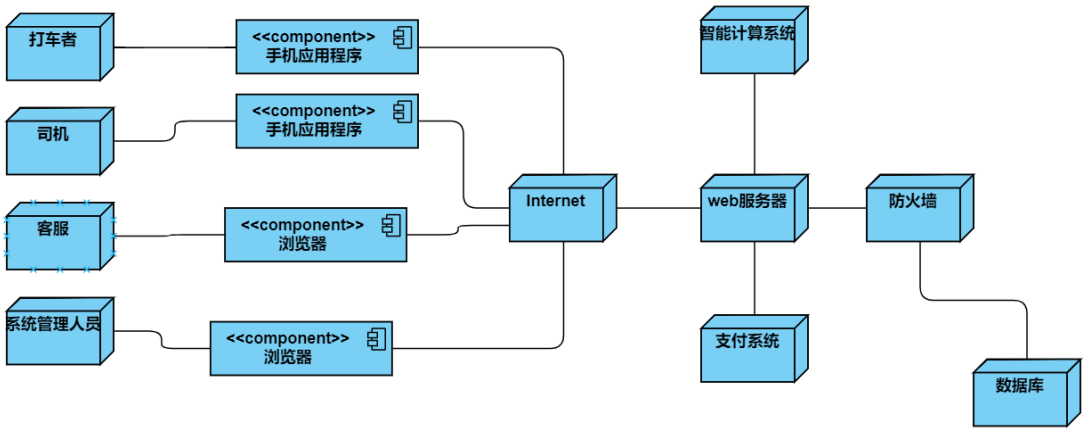


图 7 部署视图

7.1 手机应用程序

打车者和司机在智能手机上使用此打车软件，这些智能手机通过无线网络连接上服务器，并与本软件的服务器进行交互，实现功能，如发起打车订单，查询订单，接单，支付等。

7.2 浏览器

管理员，客服在地台式机上使用此软件，这些台式机通过 LAN 直接连接到服务器。管理员可以通过台式电脑在此软件上

7.3Internet

注册服务器是主要的园区 UNIX 服务器。 所有的教职员工都可以通过校园局域网访问服务器。

7.4 智能计算系统

课程目录系统是一个包含完整课程目录的遗留系统。 可通过 College Server 和 LAN 访问它。

7.5 支付系统

记帐系统（也称为财务系统）是一个遗留系统，每个学期都会生成学生帐单。

7.6 数据库

数据库里保存了用户的个人信息，支付信息，订单信息等，为打车软件的运行提供数据。
当有需要的时候,可以通过数据操作层实现对数据库里的数据进行增删改查的操作。

7.7web 服务器

在本打车软件的体系结构设计的过程中，web 服务器的作用是为用户发出的请求提供服务。与 web 服务器相连接的有界面层，数据库，支付系统和智能计算系统。用户在使用此软件的时候，首先发出使用服务的请求，web 服务器收到用户的请求之后，与其相连接的系统完成请求并返回数据到界面层。

8. 尺寸和性能

所选择的软件体系结构中规定的关键大小和时序要求:

8.1 时间性能

1）系统的一般操作响应时间应该不超过 2 秒，系统涉及数据连接的操作响应时间应该不超过 5 秒。
2）系统支持 100000 个用户同时在线，并在软件中执行一些操作。
3）系统支持 50000 个用户同时并发登陆系统，且产生的系统延迟不超过 3 秒。当有 10000 人同时登陆时，登陆产生的系统延迟不超过 5 秒。
4）系统支持 20000 个用户在一定时间范围内同时进行叫车，而且系统能够准确的进行订单有效性的检查，并返回给用户相关的信息。
5）系统产生警告信息到将警告消息显示在屏幕上的延时不超过 5 秒。
6）系统更新数据时，最大延时为 2 小时。
7）在系统因为访问人数过多而发生宕机时，本系统应该在 5 分钟之内可以进行恢复。

8.2 空间性能：

1）本系统应该存储至少 10 万人的信息的记录，用于不同身份背景的人进行系统的使用。
2）本系统的磁盘容量应该至少为 15TB，其中应该有至少 2TB 的空闲空间，用于信息的扩充。

9. 质量

9.1 质量需求

所谓非功能性需求，是指软件产品为满足用户业务需求而必须具有且除功能需求以外的特性。非功能性需求在需求分析阶段常常被忽略或没有被足够重视。软件产品的非功能性需求包括系统的质量需求和工程需求。对于本系统，主要通过以下三个方面对本系统的非功能性需求进行描述。

9.1.2 可靠性

本打车软件应每周 7 天，每天 24 小时可用。 停机时间不得超过 1%。

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。

打车软件系统的用户数量庞大，可能会因为用户的误操作引起系统的异常。 本系统要求具有较强的容错能力，能够捕捉由于用户误操作引起的异常，并在可容忍错误 程度的情况下，保持稳定运行。

本系统要求对司机信息以及其车辆信息等关键数据做到数据的准确以及一致。同时，可以对数据使用异地容灾备份的方式，从而进一步提高打车软件系统中数据抵抗外界破坏的能力。

9.1.3 易用性

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

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，要求系统具备良好的人机交互能力。系统提供的各种功能应该便于用户理解，操作简单，用户很容易掌握。

（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。

（3）软件运行的手机界面应该为市面上主流的智能机的手机界面。

9.1.4 密安性

(1)通常来讲，实际使用的打车系统，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。

（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，以保护用户的隐私信息的安全

（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。

9.1.5 可维护性

软件的可维护性是指改进软件的难易程度。该系统的结构、接口、功能以及内部过程在开发以及跟踪阶段，容易被维护人员理解。同时，该系统有良好的测试 和诊断系统错误的功能。当该系统成为其他软件的子系统（如外卖软件）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。

9.1.6 可移植性

（1）易安装性：该系统能够跨平台移动运行，包括 安卓操作系统和 ios 操作系统。

（2）共存性：系统应当能够和其他软件共存于一个平台上，不存在冲突的软件。

（3）已替换性：系统可被容易地卸载，也易被高版本的系统替换。

9.1.7 可扩展性

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

当前制作的需求模型，可以经过修改用到共享单车，共享汽车上。

也可以经过扩展，接纳更多的司机、打车者，出租汽车公司等。

9.1.7.1 公性的软件部件

基于本体系结构的考虑，支付系统，智能计算系统，导航系统都可以形成公性的软件部件，可以将这些作为公司的资产，外卖给同行或开源。其中，支付系统和导航系统可以卖给做外卖软件的公司，智能计算系统可以外卖给共享单车的公司。此外，本软件在经过修改和扩展，可以用于共享单车，共享汽车的使用。当需要的时候，也可以将本软件卖给这些公司。

9.2 评价

本文描述的体系结构中 B/S 模式，C/S 模式和主动式仓库模式相结合，能满足打车软件对质量的需求，并且在软件的限制条件范围之内。

结合本文的体系结构和上面定义的质量需求对本体系结构文档进行评价。根据视图，得出本体系结构具有良好的性能，可靠性，可用性，安全性，可修改性，可维护性，可扩展性和可移植性等。可以按照本文档对软件进行开发。