Grab打车软件系统

体系结构设计文档

<2.0>

<2019.12.17>

<姜玥>

<学号：2017211850>

<班号：2017211501>

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** |
| <2019-12-12> | | <版本1> | <姜玥> | <第一次修订> |
| <2019-12-17> | | <版本2> | <姜玥> | <第二次修订> |

目录

[修订记录 ii](#_Toc27578549)

[1. 介绍 1](#_Toc27578550)

[1.1 目的 1](#_Toc27578551)

[1.2 范围 1](#_Toc27578552)

[1.3 定义，首字母缩略字及缩写 2](#_Toc27578553)

[1.4 参考 3](#_Toc27578554)

[1.5 概述 3](#_Toc27578555)

[2. 项目相关信息 5](#_Toc27578556)

[2.1 产品描述 5](#_Toc27578557)

[2.1.1 项目概述 5](#_Toc27578558)

[2.1.2 用户角色 5](#_Toc27578559)

[2.2 产品功能 6](#_Toc27578560)

[2.3 用例视图 10](#_Toc27578561)

[3 体系结构目标和约束 18](#_Toc27578562)

[3.1 体系结构定义 18](#_Toc27578563)

[3.2 关键指标 18](#_Toc27578564)

[3.3 各相关方对体系结构的要求 19](#_Toc27578565)

[3.4约束条件 19](#_Toc27578566)

[3.5设计遵循的标准 20](#_Toc27578567)

[3.5.1 实用性原则 20](#_Toc27578568)

[3.5.2 稳定性原则 20](#_Toc27578569)

[3.5.3 复用性原则 21](#_Toc27578570)

[3.5.4 扩展性原则 21](#_Toc27578571)

[3.5.5 安全性原则 21](#_Toc27578572)

[3.6 非功能需求 21](#_Toc27578573)

[3.6.1 性能 21](#_Toc27578574)

[3.6.2可靠性 21](#_Toc27578575)

[3.6.3 可用性 22](#_Toc27578576)

[3.6.4密安性 22](#_Toc27578577)

[3.6.5 可维护性 22](#_Toc27578578)

[3.6.6可移植性 23](#_Toc27578579)

[3.6.7可扩展性 23](#_Toc27578580)

[3.7 工程要求 23](#_Toc27578581)

[3.7.1 设计约束 23](#_Toc27578582)

[3.7.2 逻辑数据库要求 23](#_Toc27578583)

[3.8 其他需求 24](#_Toc27578584)

[3.8.1 界面需求 24](#_Toc27578585)

[3.8.2 数据容量的需求 24](#_Toc27578586)

[4 现有三层B/S体系结构分析 25](#_Toc27578587)

[4.1 三层B/S体系结构概述 25](#_Toc27578588)

[4.2 三层B/S体系结构的优点 26](#_Toc27578589)

[4.3 三层B/S体系结构对质量属性的影响 27](#_Toc27578590)

[5.体系结构 29](#_Toc27578591)

[5.1 应用B/S 体系结构 29](#_Toc27578592)

[5.1.1 概念级体系结构 29](#_Toc27578593)

[5.1.2 模块级体系结构 30](#_Toc27578594)

[5.1.3 运行级体系结构 31](#_Toc27578595)

[5.1.4 代码级体系结构 32](#_Toc27578596)

[5.2 结构化视图 34](#_Toc27578597)

[5.3 逻辑视图 34](#_Toc27578598)

[5.3.1  Architecture Overview – Package and Subsystem Layering 35](#_Toc27578599)

[5.4过程视图 36](#_Toc27578600)

[5.1 Processes 36](#_Toc27578601)

[5.2 Process to Design Elements 37](#_Toc27578602)

[6.3 Process Model to Design Model Dependencies 38](#_Toc27578603)

[6.4 Processes to the Implementation 39](#_Toc27578604)

[5.5部署视图 40](#_Toc27578605)

[6. 系统的质量分析和评价 42](#_Toc27578606)

[6.1 场景分析 42](#_Toc27578607)

[6.1.1 用例场景 42](#_Toc27578608)

[6.1.2 增长性场景 42](#_Toc27578609)

[6.1.3 探索性场景 42](#_Toc27578610)

[6.2 原型分析 42](#_Toc27578611)

[6.3 风险 44](#_Toc27578612)

[6.3.1 技术风险 44](#_Toc27578613)

[6.3.2 进度风险 44](#_Toc27578614)

[6.3.3 质量风险 45](#_Toc27578615)

1. 介绍

在这一部分，主要介绍了本体系结构文档编写的目的、范围、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），分析现有的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。

本文档预期读者对象为：

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. 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，首字母缩略字及缩写

|  |  |
| --- | --- |
| Grab打车软件系统 | 一个分别面向司机、乘客、系统人员，通过乘客叫车，给司机派单，司机载客，系统人员监督方式进行的打车软件系统 |
| 用户角色 | 用户角色是指按照一定参考体系划分的用户类型，是能够代表某种用户特征、便于统一描述的众多用户个体的集合 |
| 功能需求 | 功能需求是对未来软件的服务功能进行的描述。功能需求更关注系统的输入、加工（业务流程）直到输出的情况 |
| 非功能需求 | 非功能需求是指功能以外的需求描述，主要是系统的一些关键的特性，例如可靠性、响应时间、存储空间、I/O设备的吞吐量、接口的数据格式等 |
| 用例图 | 用例图是指由参与者（Actor）、用例（Use Case）以及它们之间的关系构成的用于描述系统功能的视图。用例图（User Case）是被称为参与者的外部用户所能观察到的系统功能的模型图，呈现了一些参与者和一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。 |
| C/S结构 | C/S 架构是一种典型的两层架构，其全程是Client/Server，即客户端服务器端架构，其客户端包含一个或多个在用户的电脑上运行的程序，而服务器端有两种，一种是数据库服务器端，客户端通过数据库连接访问服务器端的数据；另一种是Socket服务器端，服务器端的程序通过Socket与客户端的程序通信。 |
| B/S结构 | B/S架构的全称为Browser/Server，即浏览器/服务器结构。Browser指的是Web浏览器，极少数事务逻辑在前端实现，但主要事务逻辑在服务器端实现，Browser客户端，WebApp服务器端和DB端构成所谓的三层架构。B/S架构的系统无须特别安装，只有Web浏览器即可。 |
| ATAM | Architecture Tradeoff Analysis Method(构架权衡分析方法），它是评价软件构架的一种综合全面的方法。这种方法不仅可以揭示出构架满足特定质量目标的情况，而且（因为它认识到了构架决策会影响多个质量属性）可以使我们更清楚地认识到质量目标之间的联系——即如何权衡诸多质量目标。 |

1.4 参考

[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 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

2.1.1 项目概述

此打车软件系统，是一个基于Web的数据库应用系统，通过智能分析极大地提高生活中日常出行的效率，解决在打车出行这一方面改变传统的接单方式、保障每次打车的费用成功支付、以及保证乘客最优的打车体验等问题。本文档试图从总体架构上给出整个系统的描述，

基本的业务流程：打车软件系统可以即时对运行空车信息进行汇总，在乘客提交打车请求后，通过智能化和信息化的方式结合地理位置的方式、以多种策略并行（如红包、调度费、司机抢单等）给空车司机进行派单，司机得到系统派单后根据智能导航接送乘客，乘客到达目的地后，通过软件进行线上支付。打车软件系统实时线上监测各项数据进行智能管理，监控安全问题、智能调配，提升各项效率。

2.1.2 用户角色

在Grab打车软件中由三类用户：司机、乘客以及平台客服人员。这三类用户角色由不同的系统使用权限，可以通过软件进行不同的信息查询和处理的操作。不同的用户角色对系统使用的需求不同，具体如下：

1. 乘客交互模块：

乘客交互模块是针对乘客用户设计的，主要思想如下：

1. 系统要求乘客用户在进入系统之前输入用户名和密码进行信息验证，要求用户在第一次使用系统进行叫车之前至少绑定一种支付方式，至少绑定一个手机号以实现线上扣费。
2. 系统要求当乘客用户密码校验失败时，会提示错误，如果超过系统允许的最大连续出错值，就对该用户的账号进行锁定，提示用户通过安全验证找回账号。
3. 系统应能够实现乘客用户可以根据自己的喜好来修改个人信息，更改绑定的手机号，更改密码，或者更改绑定支付方式。
4. 系统应该实现乘客的叫车功能、预约叫车功能，实时显示车费功能，评价功能，以及查看历史订单的功能。
5. 司机交互模块：

司机交互模块是针对司机用户设计的，主要思想如下：

1. 系统要求司机用户在进入系统之前输入用户名和密码进行身份验证，要求司机进行机动车驾驶证验证和驾驶车辆信息的验证，审核通过后才能使用本系统接单；要求司机进行实名身份验证。
2. 系统要求当密码校验失败时，提示错误信息，当超过系统允许的最大连续出错值，就对该司机用户的账户进行锁定，提示用户通过安全验证找回账号。
3. 系统应该能对在线司机进行定位，并且将附近的乘客提供的订单信息智能分配，提供相关行程信息。
4. 系统应该实现司机用户根据个人喜好来修改个人信息，更改绑定的个人支付账号等。
5. 平台客服人员交互模块

平台客服交互模块主要针对打车软件管理平台客服人员设计的，主要设计思想如下：

1. 系统要求平台客服人员在进入系统之前输入用户名和密码进行身份验证。
2. 系统要求当密码验证失败时，会提示错误，当超过系统允许的最大连续出错值，那么会对该用户进行锁定。
3. 系统能实现提供客服人员查询乘客、用户和行程部分信息的权限，客服人员可以处理乘客投诉信息、司机投诉信息，查询相关行程信息反馈给用户。
   1. 产品功能

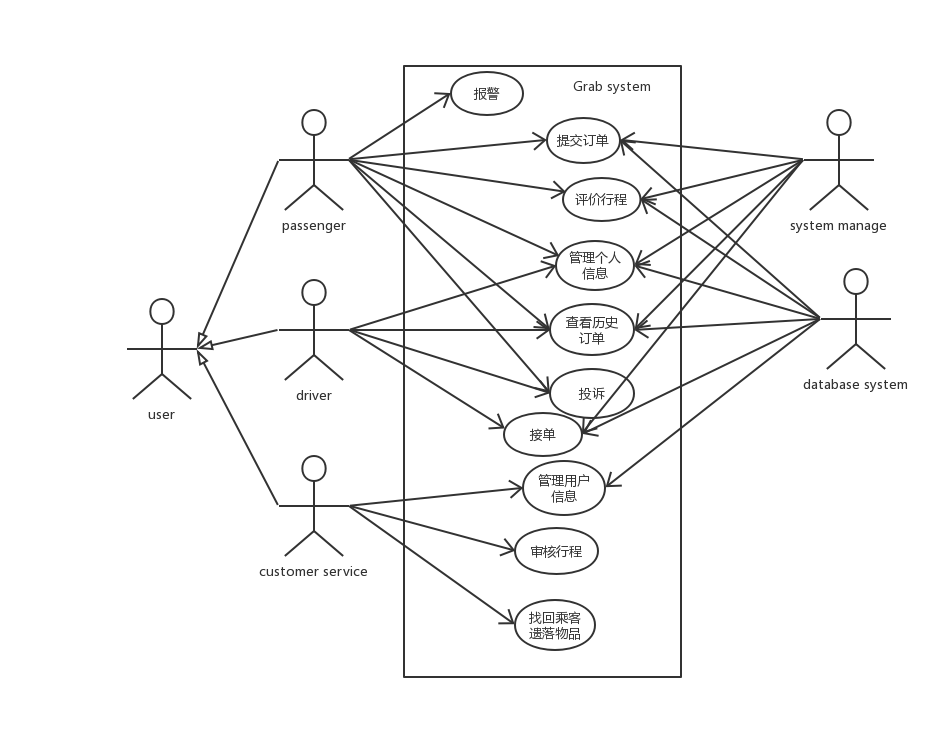
|  |  |
| --- | --- |
| 用户 | 功能 |
| 乘客 | 乘客 注册登录  乘客注册登陆账户、在线上完成个人实名认证 |
| 乘客 实时叫车  乘客叫车下单，通过定位，呼叫司机接单，司机可以看到附近打车信息 |
| 乘客支付订单  支付功能，乘客可以选择线下支付，或者跳转第三方支付页面，或者选择好友或亲人代付 |
| 乘客联系司机  乘客可以通过软件线上联系司机，或通过APP拨打司机隐私号码进行联系 |
| 乘客选择叫车类型  乘客可选择呼叫车的类型，可以看到不同类型车的预计价钱 |
| 乘客 安全管理  保障乘客安全，一键报警，添加紧急联系人，将行程分享给好友 |
| 乘客预约叫车  乘客可以预约叫车，选择预计乘车时间，呼叫司机接单，在预约时间和地点乘车 |
| 乘客评价  乘客可以对已完成的行程和司机评价，投诉与反馈 |
| 乘客呼叫代驾  乘客可以呼叫代驾， 实时或者预约代驾 |
| 乘客 遗落物品找回  乘客发现遗落物品在车上时，可以呼叫软件客服，客服核实乘客和行程信息，提供司机联系方式给乘客，并且联系司机、查找行程中的录音信息。 |

|  |  |
| --- | --- |
| 司机 | 司机接单  乘客提交打车申请，在软件计算范围内的司机会全部接到乘客起点和目的地和信息的广播，想要接单的司机操作选择接单，之后提供乘客的详细定位信息和联系信息。一旦接单不予取消订单。 |
| 司机 查看订单  乘客提交预约以及代驾信息，司机可以查看乘客和订单详情信息 |
| 司机 查看路径  在行程中司机可以查看路径规划信息和导航信息，可以选择不同的路径，并且提供道路信息、预计行程时间和预计价格 |
| 司机 公共电台  本地司机之间可以在公共电台中分享实时道路信息。 |
| 司机 安全保证  司机可以选择一键报警，设置紧急联系人，保证司机的安全 |
| 司机 投诉反馈  在行程结束后，如果未能及时收到乘客的线上支付，可以投诉乘客，反馈给平台，由平台对用户进行提醒和督促。 |
| 司机 查看评价  司机在行程结束后查看用户的评价。 |

|  |  |
| --- | --- |
| 平台 | 软件 分配订单  平台根据司机和乘客的定位信息将乘客的叫车申请广播给最合理范围内的所有司机，司机先接先得，并且综合评价更好的司机有优先级。 |
| 软件 地图接口  软件使用第三方地图显示位置信息、路况信息和本地服务信息，显示行程的导航信息。 |
| 软件 支付接口  软件使用网银、支付宝、微信等第三方支付公司。 |
| 平台 客服  软件对行程中的录音信息保留5天，以便于行程过后乘客遗落物品产生的纠纷提供证据。 |

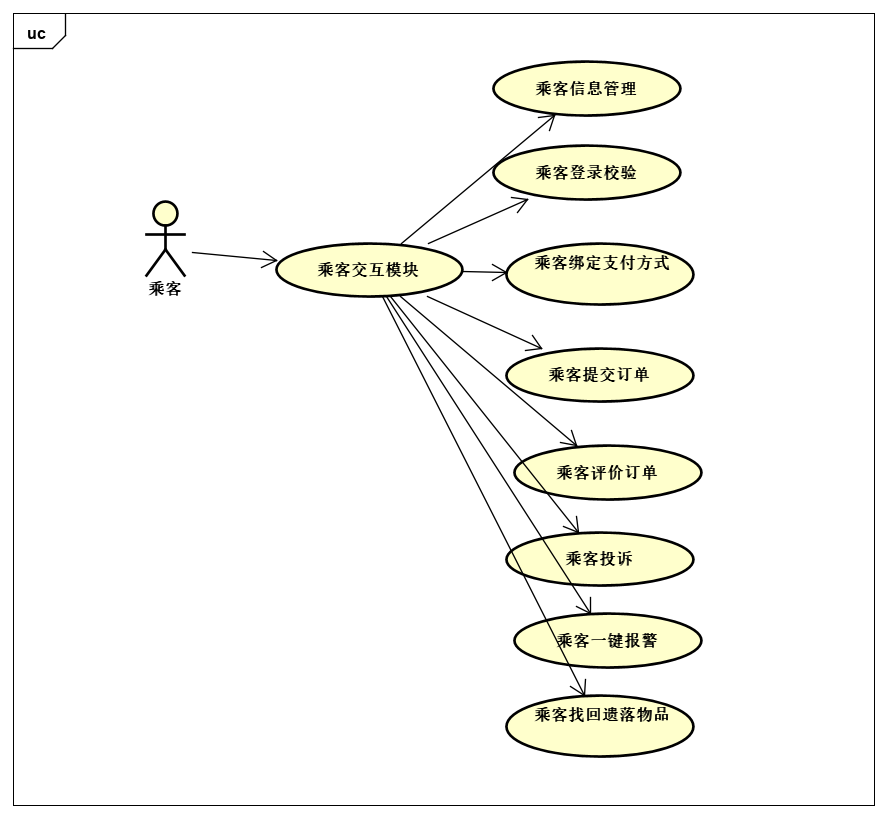
2.3 用例视图

打车软件共有五种角色：乘客、司机、平台客服人员、系统管理人员、数据库



**图2 Grab打车系统用例图**

1. 乘客-用例图1

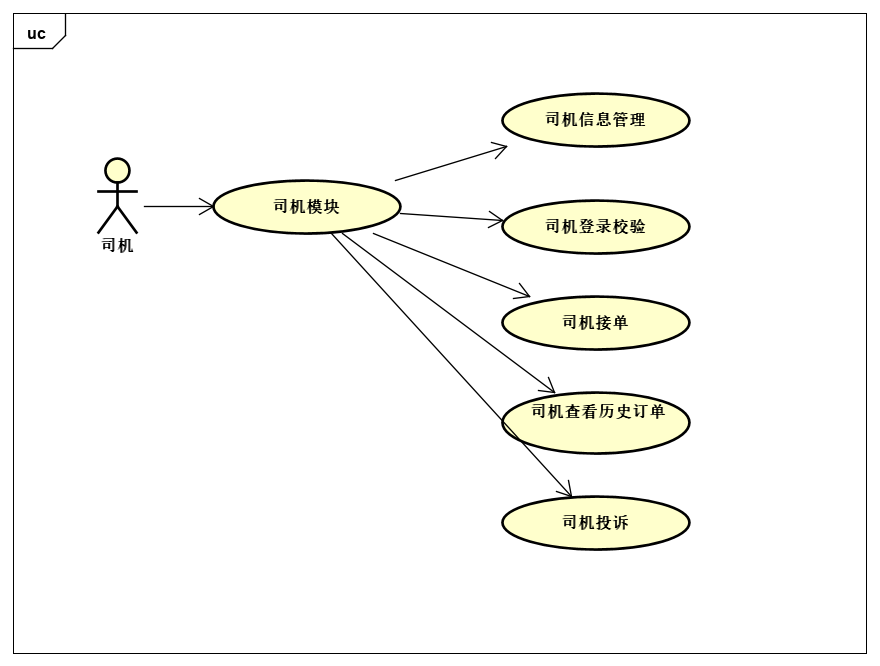


**图3 乘客用例图**

主要文本用例：

|  |
| --- |
| 用例名称：提交订单 |
| 范围：打车软件系统 |
| 级别：用户目标级别 |
| 主要参与者：乘客 |
| 涉众及其关注点：  -司机：乘客是提交订单信息准确，是否可以接单，是否需要联系乘客  -客服：乘客乘车安全是否可保证 |
| 前置条件：乘客成功登录软件系统，乘客所在地区有足够多的司机注册使用本软件，乘客账户绑定支付账户，乘客打开定位，乘客输入目的地明确 |
| 成功保证：乘客成功提交订单，等待有限时间后有司机接单并联系乘客 |
| 主成功场景：   1. 乘客打开软件和定位 2. 乘客输入起点、目的地和用车时间 3. 乘客查看预计路上的时间和车费 4. 乘客查看附近可用车辆数 5. 乘客提交用车申请 6. 乘客查看预计等待时间 7. 乘客收到订单成功接单信息，收到司机车牌号和司机用户信息 |

1. 司机-用例图2

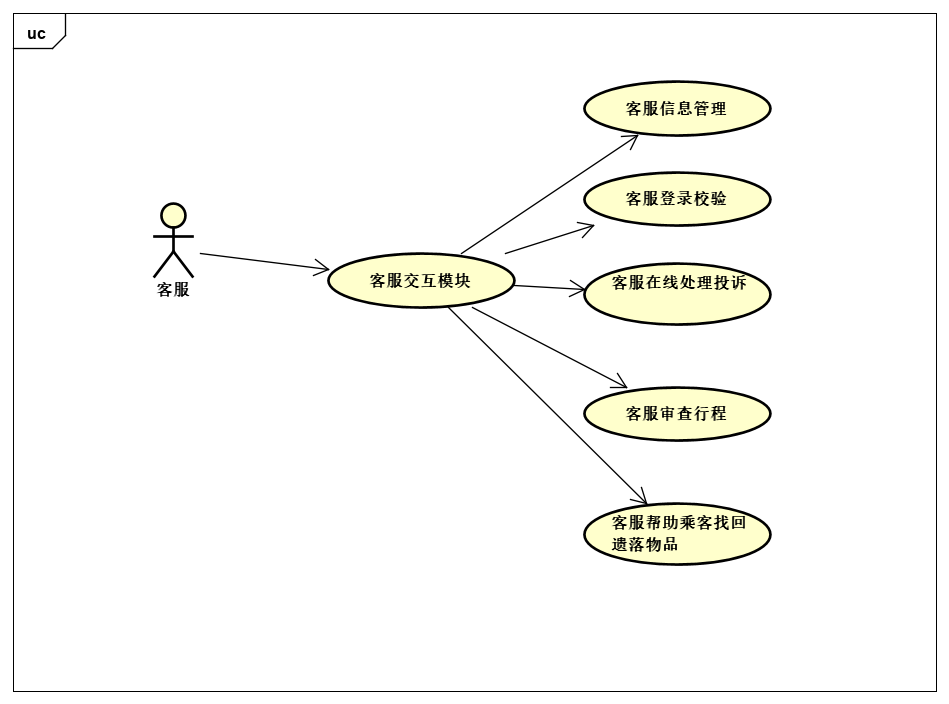


**图4 司机用例图**

主要文本用例：

|  |
| --- |
| 用例名称：司机接单 |
| 范围：打车软件系统 |
| 级别：用户目标级别 |
| 主参与者：司机 |
| 涉众及其关注点：  -乘客：司机接单是否及时，司机是否能按时到达目的地  -客服：司机是否能顺利接单 |
| 前置条件：司机成功登录软件，司机打开手机定位，司机通过平台的身份验证。 |
| 成功保证：司机成功接单，收到乘客的定位信息和乘客加密联系方式 |
| 主成功场景：   * + 1. 司机打开软件和定位     2. 平台根据司机的位置信息，计算附近可派发给司机的订单     3. 软件通过语音提醒订单的时间、起点和终点     4. 司机选择接单     5. 接收订单后司机收到乘客的起点具体位置、乘客的加密联系方式以及从司机现在位置到乘客位置的道路导航信息。 |

1. 平台客服人员-用例图3

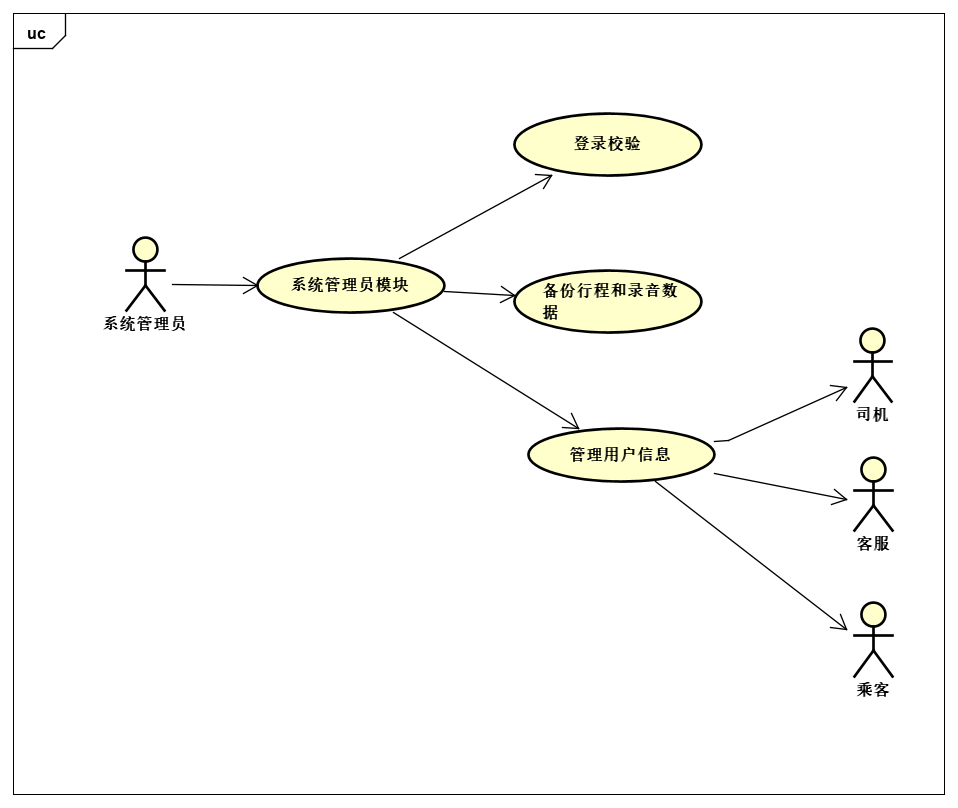


**图5 客服用例图**

主要用例的用例文本：

|  |
| --- |
| 用例名称：帮助乘客找回遗落物品 |
| 范围：打车软件系统 |
| 级别：用户目标级别 |
| 主要参与者：客服 |
| 涉众及其关注点：  -乘客：物品是否遗落在车上，是否能联系到司机  -司机：物品是否遗落在车上 |
| 前置条件：乘客遗落物品在车上，司机拾到遗落物品 |
| 成功保证：乘客与司机联系，乘客成功找到遗落物品 |
| 主成功场景：   1. 乘客联系客服表示物品遗落 2. 客服验证乘客身份信息 3. 客服验证行程信息 4. 客服找到对应行程的司机联系方式 5. 客服反馈给乘客司机的联系信息 |

1. 系统管理员-用例图4



**图6 管理员用例图**

主要用例的用例文本：

|  |
| --- |
| 用例名称：管理用户信息 |
| 范围：打车软件系统 |
| 级别：用户目标级别 |
| 主要参与者：系统管理员 |
| 涉众及其关注点：  -乘客：个人信息是否会被泄露，在行程后出现问题是否还能找到行程相关信息  -司机：个人身份信息是否会被泄露，车辆信息是否保证安全，行程出现问题后是否能找到行程相关信息。 |
| 前置条件：系统管理员登陆打车软件系统，行程信息正确存储 |
| 成功保证：系统管理员查看并管理权限内的用户个人信息 |
| 主成功场景：   1. 系统管理员登陆打车软件系统 2. 系统管理员可根据用户手机号等信息查找出相应用户信息 3. 系统管理员可根据行程时间、行程起始地址乘客信息和用户信息查找到对应的行程信息和行程中的安全记录信息。 4. 系统管理员可以删除修改用户（司机、乘客）信息 5. 系统管理员可以增加、修改、删除平台客服人员信息。 |

3 体系结构目标和约束

3.1 体系结构定义

软件体系结构是具有一定形式的结构化元素，即构件的集合，包括处理构件、数据构件和连接构件。处理构件负责对数据进行加工，数据构件是被加工的信息，连接构件把体系结构的不同部分组组合连接起来。这一定义注重区分处理构件、数据构件和连接构件，这一方法在其他的定义和方法中基本上得到保持。由于软件系统具有的一些共通特性，这种模型可以在多个系统之间传递，特别是可以应用到具有相似质量属性和功能需求的系统中，并能够促进大规模软件的系统级复用。

## 3.2 关键指标

本系统主要面向乘客、司机，使用本系统，乘客可以随时随地进行叫车，并且获得当前到目的地的时间和车费；司机可以高效率的接单，节省时间同时获得更多利益，司机和乘客可以通过本系统软件在个人信息受保护的情况下进行交流，双方的位置、搜集好等收到保护，同时，由于所储存的司机及车辆的信息、以及乘客的信息十分重要，在行程中司机和乘客的安全需要得到保证，所以要求系统具有较高的稳定性和安全性，降低系统出错率，并且能够准确对司机和乘客进行定位，平台客服人员能够第一时间查到用户此时的位置信息，实时了解行程信息，所以在性能上评价Grab打车软件需要具备以下关键指标：

系统应支持至少50万个司机和70万个乘客的使用.

系统应该支持在10万个用户同时并发登陆系统，并且系统产生的延迟不超过5秒钟。

系统将信息更新至数据库时对数据的处理量在G级别时，系统的响应时间应该在5秒内。

系统因并发访问人数过多而发生宕机时，系统应该在6分钟之内进行系统恢复，并且宕机期间的正在进行的行程信息可以通过备份查到，并且在恢复之后可以更新当前行程信息。

3.3 各相关方对体系结构的要求

Grab打车软件系统结构视角如下图所示，该模型从4个角度（逻辑、实现、进程和部署）指出不同的相关方的关注点，此外，还包括从最终用户角度对用例做观察，分析其影响系统的上下文和商业目标的情况。系统最终用户和系统设计人员要求系统可以实现需求分析阶段的基本功能，对于Grab打车系统，要求系统能满足司机、乘客、平台客服人员三种不同用户的功能需求，通过限制不同角色的使用权限，实现便捷、高效、安全有保障的打车软件系统。

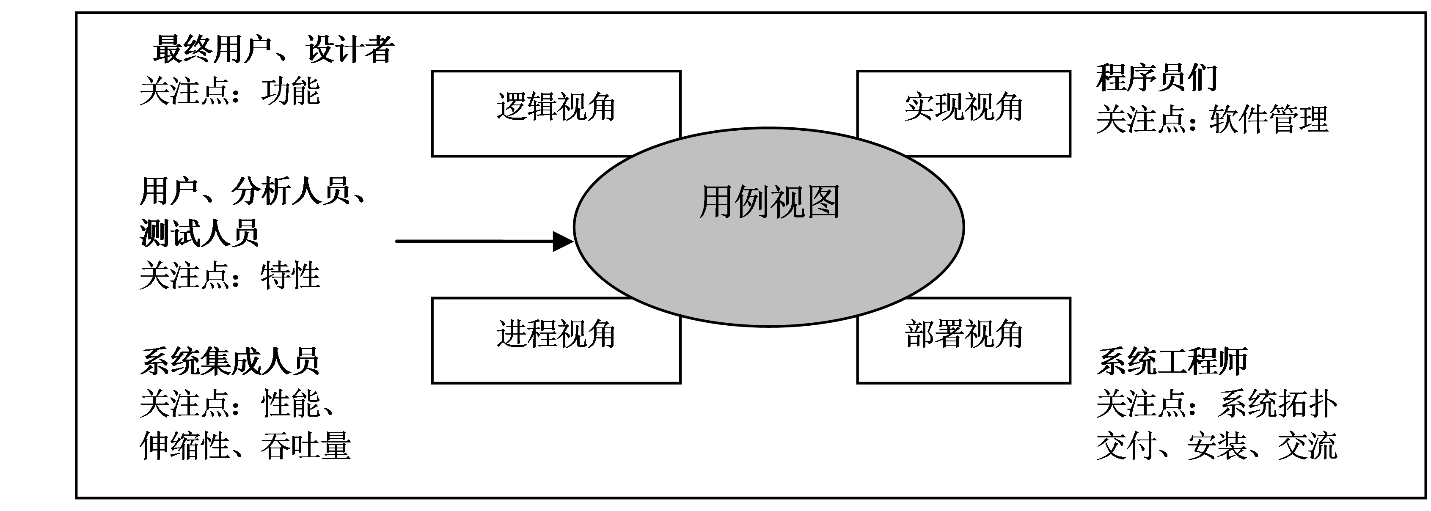


图7 软件体系结构视角

1. 最终用户和软件分析人员、软件测试人员要求系统的体系结构能满足需求分析的功能，要求系统可靠正确运行，数据安全完整，界面简洁，可操作性强且用户体验良好。
   * 1. 系统开发人员要求体系结构设计合理，方便进行修改、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宠用，方便形成模板框架来进行其他系统的二次开发。
     2. 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，比如系统时间性能和空间性能的要求。要求系统各个模块和层次之间尽量松耦合以方便哥哥部件之间的集成。
     3. 项目工程师要求系统在规定时间内完成并交付，要求软件系统体系结构能完成规模和时间sha能够的要求，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。

## 3.4约束条件

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。

1. 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。
2. 程序启动后，有合理的用户界面，能实现人机交互的功能。
3. 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。
4. 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。
5. 项目基本限制：
6. 本打车软件的性能必须能满足100万并发打车，1亿名用户，1000万名司机使用。
7. 项目技术上的限制：
   1. 常用的信息管理系统采用 B/S 开发模式，系统应采用网站开发的基本技术。
   2. Web 服务器部署于 Linux 系统之上的系统一般运行稳定性高。
   3. 服务器系统硬件的配置能支持服务器高效稳定的运行。
   4. 为了系统将来的可扩展性，系统在硬件的使用上需尽可能减少针对性。整个系统也应尽量减少各模块间的调用，尽量做到松耦合。
   5. 数据文件、系统配置文件应当安全可靠的存储。系统的数据需要具有足够的可靠性，才会保证系统正常运行

3.5设计遵循的标准

该软件系统必须遵循以下设计原则：

3.5.1 实用性原则

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

3.5.2 稳定性原则

打车系统提供的 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。

3.5.3 复用性原则

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

3.5.4 扩展性原则

打车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。

3.5.5 安全性原则

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

* 1. 非功能需求

### 性能

1. 系统执行速度：系统执行速度需要满足以下要求：系统响应块，数据库单表操作时间不大于0.5秒
2. 系统响应时间考虑到同时进行操作的用户人数，系统的响应时间应该达到：系统应当可以查看和修改用户的基本信息。一个流程的响应速度最佳为1 s,平均为1 s，不超过1 s。系统应当可以同时接受 1000000人同时操作。
3. 系统应当可以查询在线司机信息，一个流程响应素服最佳为2s，平均为2s，不超过2s。
4. 系统应当能准确更新行程中的定位信息，道路规划信息，和实时车费计价信息。一个流程的响应速度最佳为 b3s,平均为 a3 s，不超过 m3s。系统应当可以同时接受 10000 人同时操 作。
5. 系统应该能计算最佳司机分配订单，一个流程的响应速度最佳为 b4s,平均为 a4s，不超过 m4s。系统应 当可以同时接受 10000 人同时操作。
6. 空间性能：系统应该可以查看修改用户信息、查询行程订单和录音数据、更新行程中的定位信息、计算最佳分配订单方式，每次操作占用内存不超过s3.

### 3.6.2可靠性

1. 系统能较长时间下稳定运行，同时该系统需要具备一定的故障恢复能力，即有一定的容错能力。当用户的操作不当引起某些故障时，或者是由于操作系统或者网络发生故障时，系统需要具备一定的故障恢复能力。
2. 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

### 3.6.3 可用性

1. 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 该具备容易操作的功能。 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。 系统应提供在线帮助界面，方便用户学习操作。
2. 由于操作该系统的人员有很多，且操作习惯、受教育程度、年龄阶段、接受 事 物能力等都各不相同，这就要求系统具备良好的人机交互能力。系统提供的各 种功能便于用户理解，操作简单，用户很容易掌握。
3. 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。
4. 系统应具有一定的美观性，可参考目前大部分网站的扁平化设计。

### 3.6.4密安性

1. 通常来讲，实际使用的管理系统，必须具备相应的安全性能。该系统各级用户有各自的权限设置，例如用户之间不可以互相查看或修改其他人的个人信息。
2. （2）该系统应该通过设置防火墙确保数据传输的安全。使用可靠的操作系统来保证系统的操作安全，确保系统在一个安全可靠的环境下运行。
3. 系统应保证用户信息不泄露，系统配置文件和数据库存储文件应当进行加密处理。
4. 系统应保证不会因恶意攻击而崩溃，系统开发过程不存在明显漏洞。
5. 系统应当能够保证选取的开发方不存在商业竞争对手或类似的恶意对手。

### 3.6.5 可维护性

* + - 1. 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。
      2. 系统应该能够容易诊断出存在的缺陷和失效原因，容易识别出待修改部分的可能性或能力。开发人员应当记录开发过程日志，以便备份追踪。
      3. 系统应当能够保证开发过程的代码、设计和文档容易修改，代码应当结构清晰且有较详细的注释，设计文档详细明确。

### 3.6.6可移植性

1. 系统应当能够保证在不同的手机操作系统上容易安装和部署。
2. 系统应当能够和其他软件共存于一个平台上，存在冲突的软件不超过。系统应 当能够容易地被卸载，也容易被更高版本的系统替换。

### 3.6.7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。

## 3.7 工程要求

### 3.7.1 设计约束

1. 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。
2. 系统要求服务器采用Java语言进行编写，客户端采用.NET开发技术，并且所有变量的命名规范符合相应语言命名规范。
3. 本系统手机端应用框架采用Spring Boot框架进行系统开发，网页端使用JSP技术来实现。
4. 本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。

### 3.7.2 逻辑数据库要求

1. 要求使用关系型数据库来实现数据的存储与管理。系统推荐使用 MySQL。若经费预算允许，可使用大型数据库，可靠性高，稳定性好；
2. 要求数据库中必须存在乘客用户、司机用户、平台客服人员三种实体，并且使用不同的主键来标识不同的实体。
3. 要求数据库的访问操作要有相应的并发处理和恢复机制，保证在大量访问的情况下获取数据的正确性。
4. 数据库表的字段类型定义应具有一定灵活性；
5. 数据库表的数据应当具有一致性和完整性；
6. 满足上述前提下，数据库数据存储应尽量节省空间。

## 3.8 其他需求

### 3.8.1 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司机来说可以很容易理解系统的各个功能。

### 3.8.2 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。

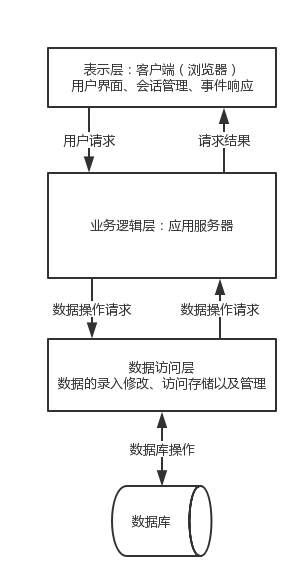
4 现有三层B/S体系结构分析

4.1 三层B/S体系结构概述

B/S结构（Browser/Server，浏览器/服务器模式），是WEB兴起后的一种网络结构模式，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。客户机上只要安装一个浏览器（Browser），如Chrome或Internet Explorer，服务器安装SQL Server、Oracle、MYSQL等数据库。浏览器通过Web Server 同数据库进行数据交互。B/S结构使数据及应用可通过不同平台、不同网络存取,与平台无关 ,共享程度高,可伸缩扩展性强,具有高度开放性和灵活性。

而三层B/S结构则是在B/S结构的基础上，在数据管理层和用户界面层之间增加了一层结构，称为中间件（Middleware）。三层体系的应用程序将业务规则、数据访问、合法性校验等工作放到了中间层进行处理。通常情况下，客户端不直接与数据库进行交互，而是通过COM/DCOM通 讯与中间层建立连接，再经由中间层与数据库进行交换。由于三层具备相应的独立性，修改一层的内容不需要修改其他两层的代码，因此三层结构能使项目结构更清楚，分工更明确，有利于后期的维护和升级。

下图是三层B/S结构的示意图：



**图 8 三层B/S系结构体示意图**

表示层（USL）：客户端运行后看到的用户界面。可以让用户输入数据，点击按钮，返回相应的数据信息。通过表示层，把业务逻辑和用户界面分离开来，简化客户端的操作。

业务逻辑层（BLL）：是软件系统实现业务逻辑与数据操作的核心部件，它主要用于接收用户的请求，并与数据库进行连接，通过SQL的方式向数据库服务器提出数据处理申请。等数据库接收请求并完成相关操作后，会把用户需要的数据提交给Web服务器，再有Web服务器讲结果发送到客户端供用户查看。业务逻辑层包含了本软件系统所有的业务逻辑的实现。

数据访问层（DAL)：该层负责接受业务逻辑层对数据库的操作请求，实现对数据库的录入修改、访问存储、更新等功能，再把得到的运行结果返回给业务逻辑层。

4.2 三层B/S体系结构的优点

1. 客户端无需安装，只需用浏览器即可访问，更加方便快捷。同时，由于客户端不再有业务逻辑，降低了步数和维护升级的成本以及实现的难度。客户端只需要提出请求，而相应的业务逻辑则都在业务逻辑层实现。客户端可以实现零维护，当对页面进行修改时，也不需要改动相应的业务逻辑。
2. BS架构可以直接放在广域网上，通过一定的权限控制实现多客户访问的目的，交互性较强。而且BS架构无需升级多个客户端，升级服务器即可。
3. 开发人员开发时只需要关注整体结构中的一层即可，大大降低了开发的难度。
4. 三层体系结构大大降低了层与层之间的依赖，当需要更新维护时，只需要用新的实现来替换原有层次的实现即可。同时分层还有利于各层逻辑的复用和整体的维护
5. 三层结构使系统具有更好地伸缩性。每个客户端不直接访问数据库，而是与业务逻辑层进行信息交互，再通过一个业务逻辑层与数据库进行数据的请求与获取，这样大大减少了与数据库建立连接资源消耗。
6. 业务逻辑层独立于客户端和数据库访问设计，提供了灵活性和可重用性。当药修改中间层的应用服务器时，不需要修改客户端的界面以及数据库访问层相应的代码，从而是得开发编程人员可以专注于核心业务逻辑的实现，大大简化了软件系统开发的难度。

4.3 三层B/S体系结构对质量属性的影响

|  |  |
| --- | --- |
| **质量属性** | **对质量的影响** |
| 性能 | 这种结构的性能已经得到了证明。关键的问题是要考虑每个服务器支持的并发线程数量、各层之间的连接速度以及数据传递的速度。对于分布式系统来说，降低了为完成每个需求所需的层与层之间的调用时间。 |
| 可靠性 | 通过中间层的缓冲，连接数据库的用户数大大减少。虽然增加了应用服务层，并不会使系统的性能和可靠性降低。因为在动态分布式计算系统中，客户端程序不必要确切指出应用服务的网络地址，如果应用服务器超负荷，通过统一的管理程序调度将请求转移到其他应用服务器上来消除瓶颈。 |
| 可用性 | Browser／ Server 用户的界面都统一在浏览器上，浏览器易于使用、界面友好，不须再学习使用其它的软件，一劳永逸 的解决了用户的使用问题。 |
| 密安性 | Browser／ Server 系统在客户机与数据库 服务器之间增加了一层 W eb 服务器，使两者不再直接相连，通过对中间层的用户编程可实现更加健全、灵活的安全机制。客户机无法直接对数据库操纵，有效地防止用户的非法 入侵。 |
| 可维护性 | Browser／ Server 系统的三部分模块各自相对独立，其中一部分模块改变时， 其它模块不受影响，应用的增加、删减、更新不影响用户个数和执行环境，系统改进变 得非常容易，且可以用不同厂家的产品来组成性能更佳的系统。 |
| 可移植性 | Browser／ Server 系统使用HTML， HTML 是数据格式的一个开放标准，目前大多数流行的软件均 支持HTML，同时 MIME技术使得Browser可访问多种格式的文件。 |
| 可伸缩性 | Browser／ Server 所采用的TCP／ IP、 HTTP 等标准都是开放的、非专用的，是经过标准化组织所确定的而非单一厂商所制定，保证了其应用的通用性和跨平台性。同时，标准化使得B／ S模式 可直接接入Internet ，具有良好的扩展性、伸缩性，可从不同厂家选择设备和服务。  多个服务器可以有备份，多个服务运行在同一个或多个不同的服务器上，使体系结构的规模可以得到很好的提升。在实际中，数据管理层往往成为系统能力的瓶颈。 |

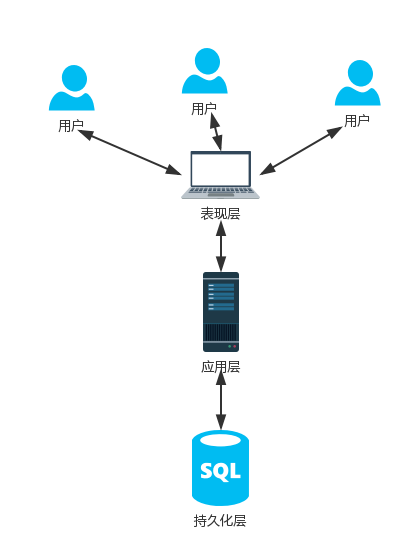
5.体系结构

5.1 应用B/S 体系结构

通过4.1部分的分析，可以知道三层B/S体系结构具有许多传统C/S体系结构不具备的优点。因此该打车系统选择基于三层B/S结构设计的开发。表现层在最外面，展现打车系统的界面，用户可以通过表现层直接与系统交互，进行一系列的操作，包括叫车，缴费，查看路线等等；中间是逻辑层，联系前端与后端；最内部是数据库访问层，用来进行数据库的相关操作。

5.1.1 概念级体系结构

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



**图 9 概念级体系结构示意图**

5.1.2 模块级体系结构

系统的模块化设计是系统进行复用的关键。模块级体系结构反应了对软件代码实现时的期望。特别是对于程序规模较大的系统。下图是按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 10 模块级体系结构**

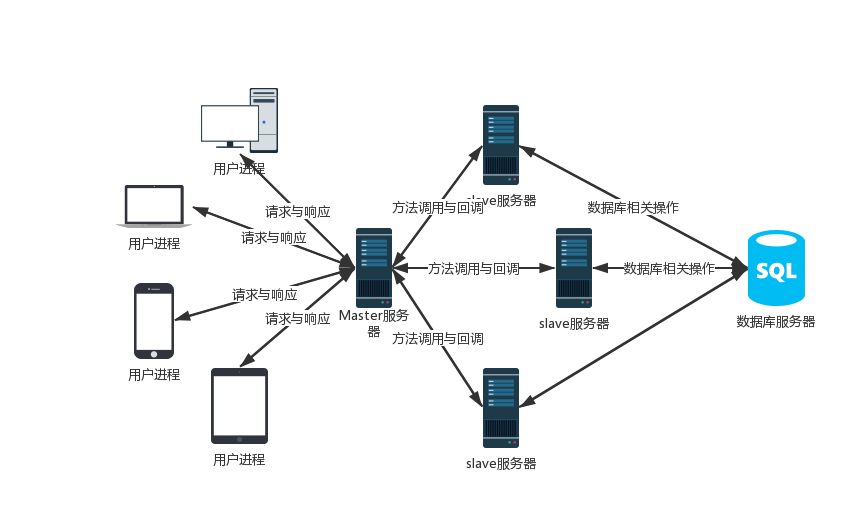
5.1.3 运行级体系结构

每个用户都是一个独立的进程，可以独立地访问服务器发送请求并查看结果。

服务器为分布式架构，一台服务器作为Master服务器，管理Slave服务器的资源配置与负载均衡。

所有的Slave均访问同一个数据库，便于维护数据的一致性，同时定期对服务器进行备份，防止可能发送的故障和奔溃。

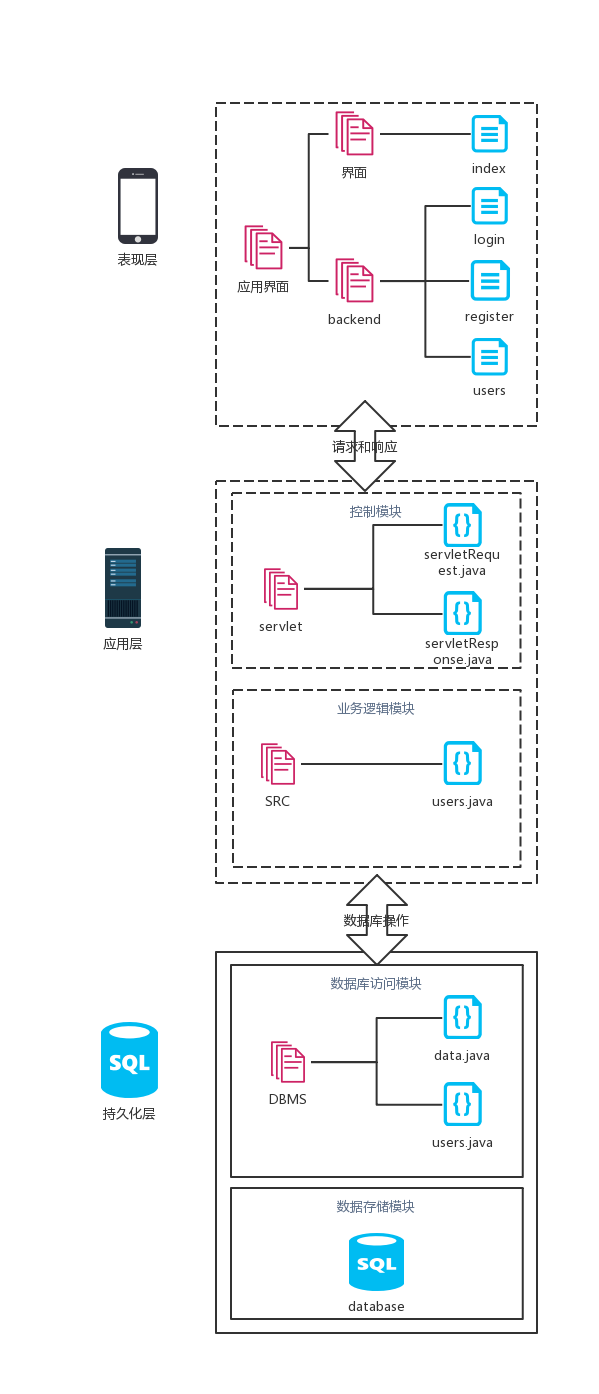
Grab打车系统的运行级体系结构如下图所示。



**图 11 运行级体系结构示意图**

5.1.4 代码级体系结构

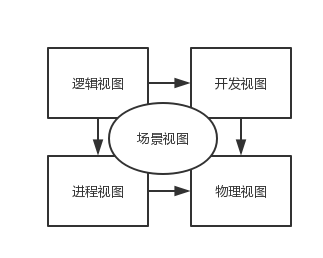
代码体系结构为编程实现提供方便，在编程语言层面按模块、目录、文件、和库的形式组织源代码。如果说在模块化体系结构设计时不要过多地考虑所采用的编程语言的话，这个阶段必须考虑编程语言、开发工具和环境(例如，配置管理)，以及项目和企业的组织结构。从体系结构设计角度看，如何降低代码之间、子项目之间的相互依赖性，提升代码的可重用性等，成为体系结构设计阶段必须考虑的问题。Grab打车系统的代码级体系结构如下图所示。



**图 12 代码级体系结构示意图**

5.2 结构化视图

Philippe Kruchten提出从不同的角度勾画系统的蓝图，建立“4+1”视图模型。该模型从4个角度（逻辑、实现、进程和部署）指出不同的相关利益方关心的事情，外加从使用者的角度对用例观察，分析其影响系统的上下文和商业目标情况。

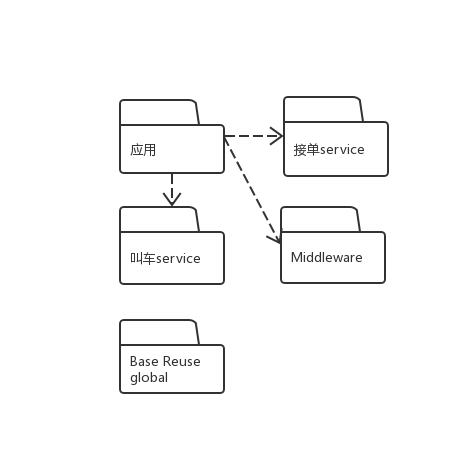


**图 13 “4+1”视图模型**

5.3 逻辑视图

逻辑视图是对体系结构的逻辑视图的描述。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。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。

### 5.3.1  Architecture Overview – Package and Subsystem Layering



**图 14 逻辑视图**

* 1. 应用层：此应用程序层具有表示用户看到的应用程序界面的所有边界类，作为用户和中间层的过度。
  2. 接单service/叫车service：此process层拥有所有的控制器类，表示驱动应用程序行为的用例管理器，此层表示用户到中间层的边界，叫车服务/接单服务依赖于流程对象层，使用户和中间层分离。
  3. Middleware：中间层支持对关系型面向对象的DBMS的访问。
  4. Base reuse：包括支持列表函数和模式的类。

5.4过程视图

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）、它们的交互和配置。还描述对象和类到任务的分配。

流程模型说明了作为可执行流程组织的课程注册类。存在支持学生注册、教授职能、注册关闭和访问外部计费系统和课程目录系统的过程。

### Processes

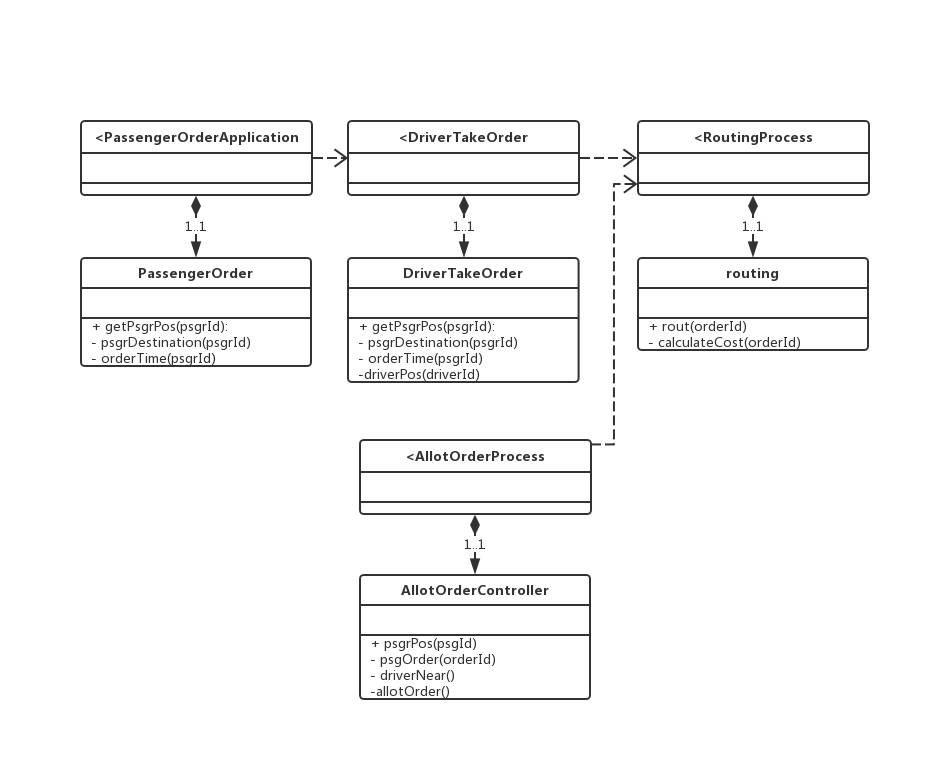


图15 过程图

1. passengerOrderApplication

此过程对应乘客提交订单的过程，获取乘客位置，获取乘客目的地信息。

1. DriverTakeOrder

此过程对应司机接单过程，获取乘客位置信息，获取乘客目的地信息，获取订单时间，获取司机位置信息。

1. AllotOrderController

此过程对应分配订单的过程，包括获取乘客位置信息，获取乘客订单，查询附近司机，智能分配订单。

1. Routing

此过程对应行程信息，规划行程路线，实时计算车费和计算预计到达时间。

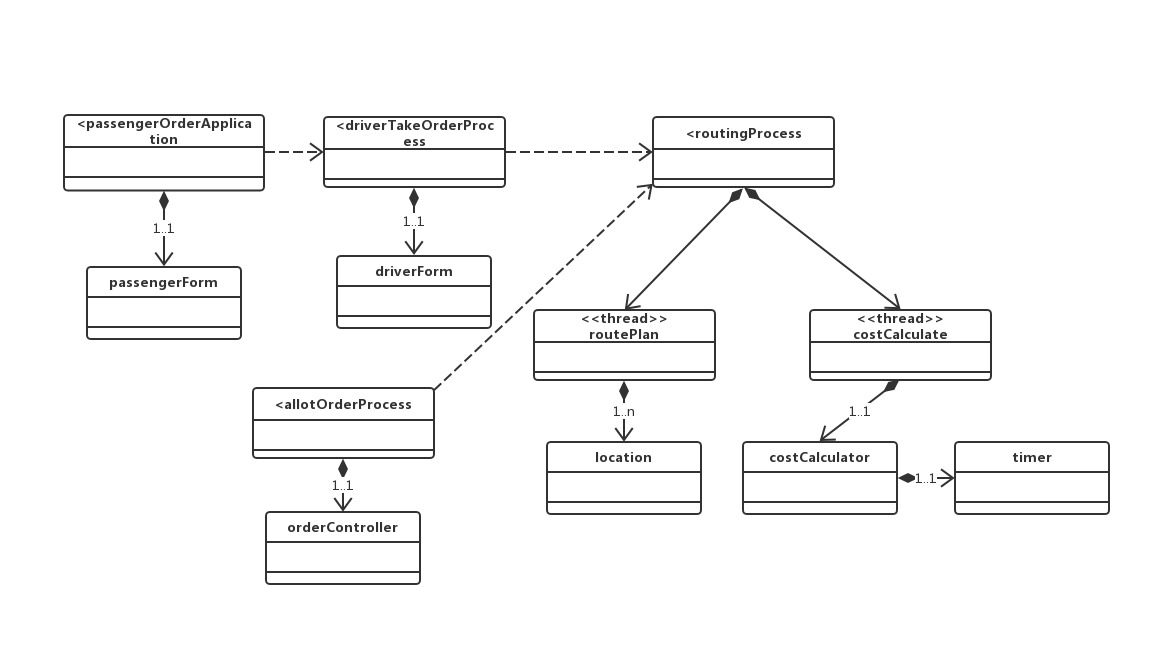
* 1. Process to Design Elements ****

图16 process to design

1. passengerOrderApplication

此过程管理乘客下订单的过程，此过程允许定位乘客现在所在的位置，获取乘客要到达的目的地址，并且根据乘客所在位置返回乘客附近的司机的位置。

1. passengerForm

对于每个正在申请叫车的乘客，此过程都有一个实例。

1. driverTakeOrderProcess

此过程管司机接单的过程，乘客提交的订单将被派发至合适的司机，提供乘客的起始位置和目的地地址，司机可以选择对派发的订单接收或拒绝，接收之后司机获得乘客的位置信息和加密的联系方式。

1. driverForm

对于每个被派发订单的司机，此过程都有一个实例。

1. allotOrderProcess

对于每一个乘客提交的订单，通过订单分配过程，计算司机位置，将订单派发给周围合适的司机。

1. routingProcess

对应处理行程的过程，其中又包括处理路线的导航过程，以及计算时间和费用的过程。

### 6.3 Process Model to Design Model Dependencies

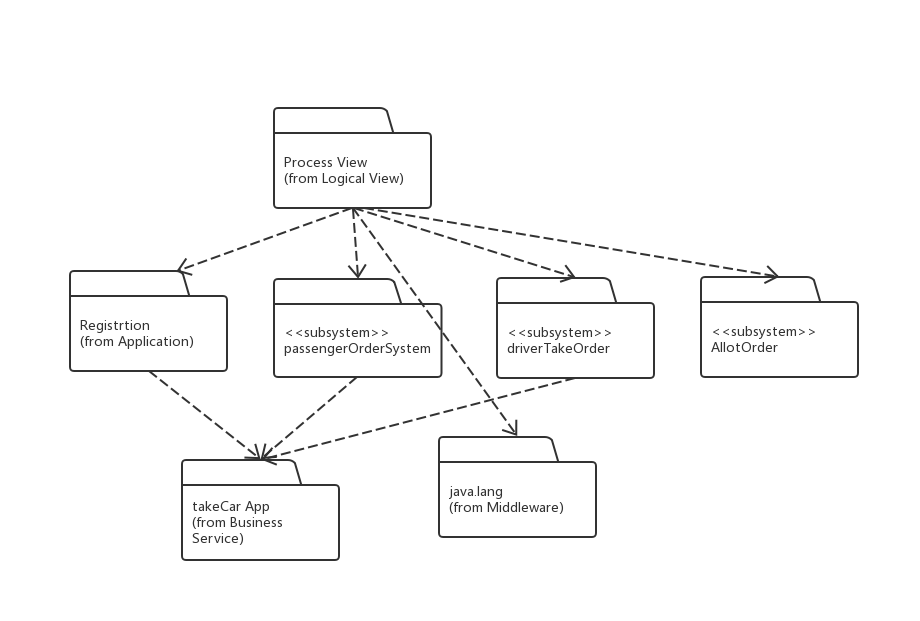


图17 process model to design Model Dependencies

### 6.4 Processes to the Implementation

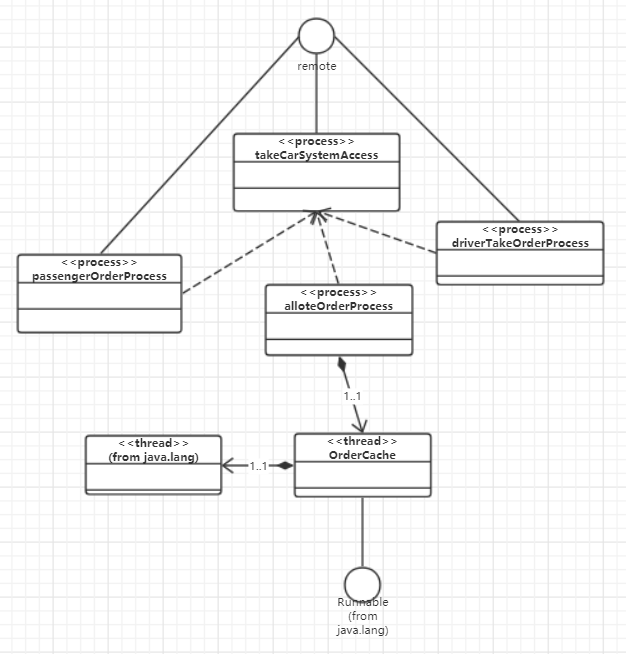


图18 Processes to the Implementation

1. Remote

远程接口用于标识所有远程对象，任何远程对象都必须直接huo 简洁实现此接口，只有远程接口中指定的方法才能远程使用。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。

1. Runnable

可运行接口应该由其实例要由线程执行的任何类实现。类必须定义一个名为run的无参数方法。此接口旨在为希望在活动时执行代码的对象提供通用协议。例如，Runnable是由类线程实现的。处于活动状态只意味着线程已启动但尚未停止

1. thread

线程是程序中的执行线程。Java虚拟机允许应用程序同时运行多个执行线程。每个线程都有优先级。优先级较高的线程优先于优先级较低的线程执行。每个线程也可以标记为守护进程，也可以不标记为守护进程。当在某个线程中运行的代码创建一个新线程对象时，新线程的优先级最初设置为等于创建线程的优先级，并且仅当创建线程是守护进程时才是守护进程线程。

5.5部署视图

架构的部署视图描述了最典型的平台配置的各种物理节点。还描述了任务（从流程视图）到物理节点的分配。

本节按物理网络配置进行组织；每个这样的配置都由一个部署图进行说明，然后是进程到每个处理器的映射

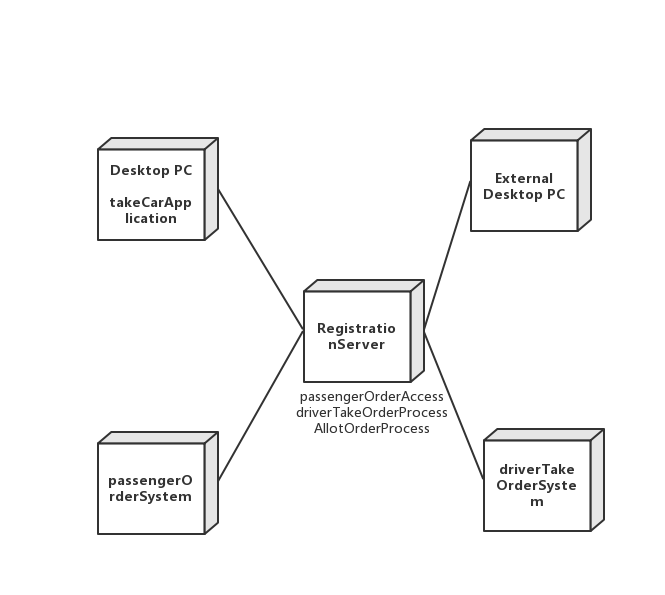


图19 deployment view

6. 系统的质量分析和评价

6.1 场景分析

根据ATAM方法来对打车系统的体系架构进行分析。

6.1.1 用例场景

1. 系统应该能够支持50000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，用户提出打车请求到系统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. 当出现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。

6.1.2 增长性场景

1. 未来会出现用户量剧增导致的服务器负载过重的问题，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. 通过对软件系统主要适配不同型号的手机或其他移动接入互联网设备，使得用户可以在各种设备终端上访问打车软件系统。

6.1.3 探索性场景

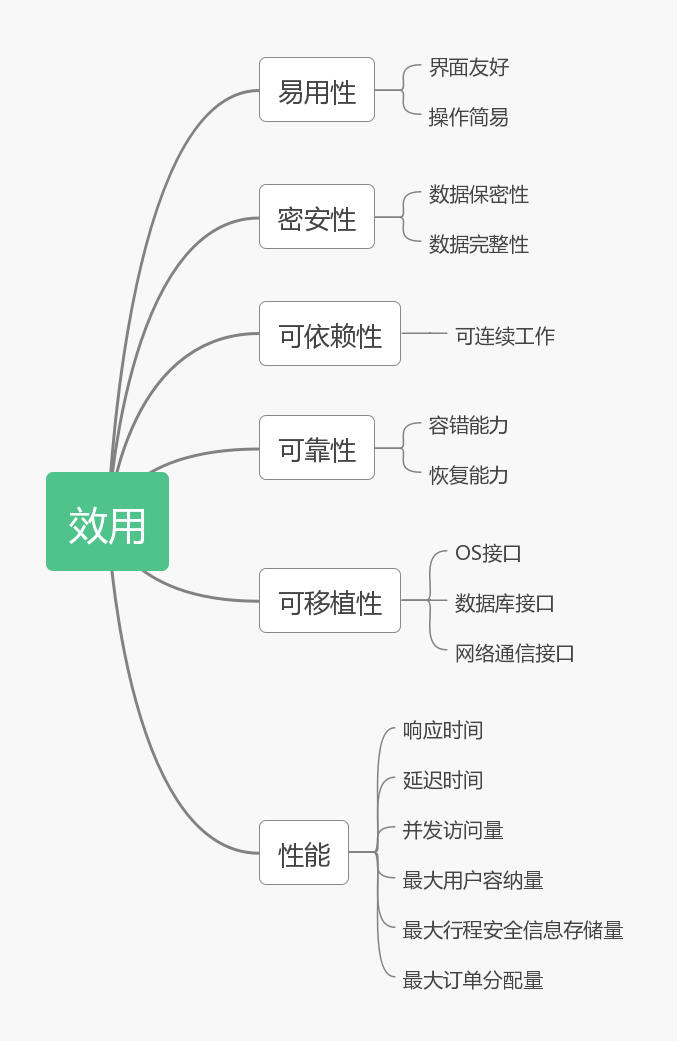
1. 通过优化算法，改进系统的性能，降低系统响应时间

2．当服务器宕机时，可以即刻启用备用服务器以至于使得用户无法感觉到服务器故障的发生，保持系统的可使用性

3. 提高系统针对高并发量访问的处理能力。

6.2 原型分析

使用效用树法进行原型分析，分析结果如下图所示



**图 20 效用树分析图**

6.3 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。

6.3.1 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。

6.3.2 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。

6.3.3 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。