智能出行车辆一体化服务管理系统

详细设计文档

小组成员：涂远鹏-1652262

刘铸煌-1652313

黎盛烜-1652310

雷成钤-1652307

指导老师：王继成

目录

1 引言

1.1 编制目的

本报告详细完成了对智能出行车辆一体化服务管理系统的详细设计，描述了系统完整且详尽的运作结构，达到指导详细设计和开发的目的，同时实现设计管理者和测试人员及用户的沟通。
本报告面向开发人员、测试人员及最终用户编写。

1.2 词汇表

词汇名称	词汇含义	备注
ITVISMS	Intelligent Travel Vehicle Integrated Service Management System	智能出行车辆一体化服务管理系统
Bmob	云端服务器	在线云端服务器
Dd	Data dictionary	数据字典
TDT	Task Decomposition Technology	任务分解技术
Dpc	Data processing cycle	数据处理流程
Dpss	Data processing system security	数据处理系统安全性
OF	Operational Feasibility	操作可行性

1.3 参考资料

1） 郭霖. 第一行代码[M]. 人民邮电出版社:郭霖, 2014. 69-307
2） IEEE[1471-2000]标准
3） 骆斌.软件工程与计算（卷二）——软件开发的技术基础[M].机械工业出版社:骆斌, 2016.82-118，403-410
4） 郭志宏. Android 应用开发详解[M].电子工业出版社. 2010.
5） 余志龙, 陈昱勋, 郑名杰, 陈小凤, 郭秩均. Google Android SDK 开发范例 大全[M]. 人民 邮电出版社. 2009.
6） 李宁. Android/OPhone 开发完全讲义[M]. 中国水利水电出版社. 2010.

2 产品概述

现有的地图软件能够提供一定程度的停车场定位服务，但是无法提供停车场内部更精确的车位信息，而我们设计的系统整合了地图定位提供的导航服务和停车场管理处的车位实时信息，相比现有系统的服务，能够节省车主在停车场内部寻找车位的时间，避免导航到停车场却发现已满的情况。

另外，目前的地图软件没有一个整合车辆服务点的平台，用户只能查找到服务点后电话联系商家或到现场咨询，浪费时间，通过这个系统能在线上完成查询-预约-导航的整个流程，节省时间成本。对于商家来说，能够在线上完成车辆服务的预约管理，合理分配服务资源。

智能车辆出行一体化系统就是为了满足上述需求而开发的系统，他包含一个数据集中服务器和若干个移动客户端，数据服务器用于将所有的用户数据集中便于维护和管理，用户则通过客户端完成相关的服务，客户端和服务端采用 TCP/IP 协议实时通信的方式完成数据的交换和维护。

3 体系结构设计概述

参考智能出行车辆一体化服务管理系统体系结构设计文档中对体系结构设计的概述。

4 结构视角

4.1 业务逻辑层的分解

业务逻辑层的开发包图参见软件体系结构设计文档。

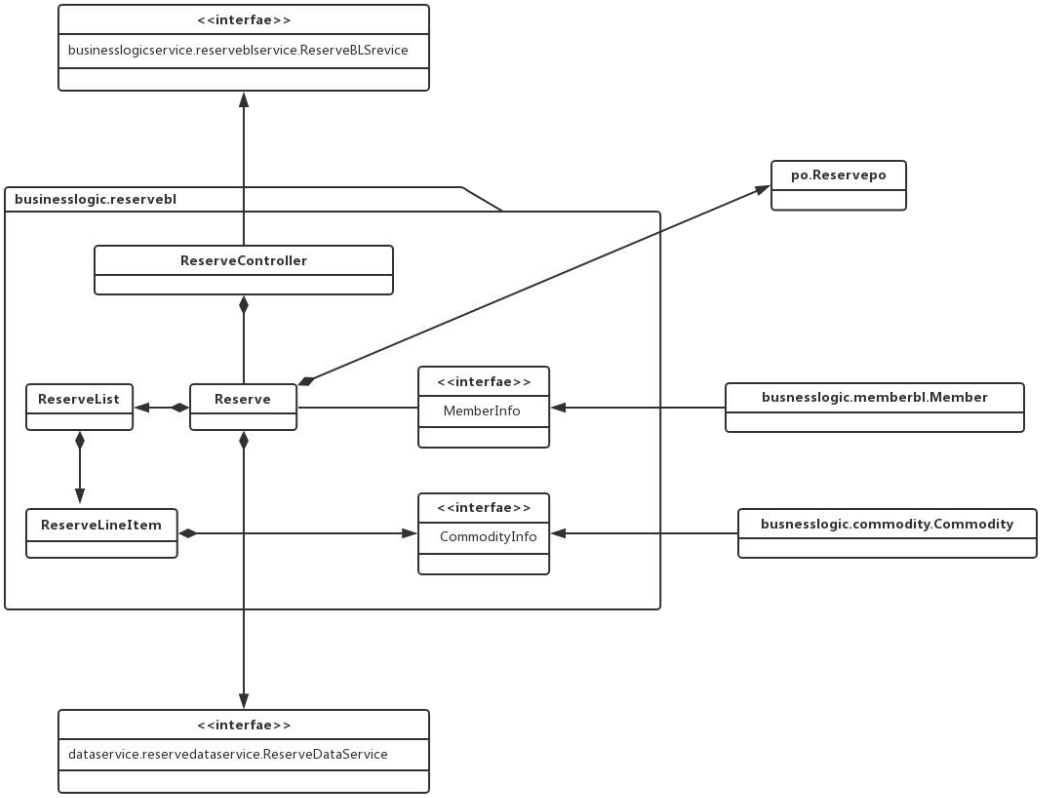
4.1.1 模块概述

Reservebl模块承担的需求参见需求规格说明文档功能需求说明和相关非功能性需求说明。

Reservebl模块的职责及接口参见软件体系结构设计文档。

4.1.2 整体结构

根据体系结构的设计，我们将系统分为用户界面层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们会添加接口。比如用户界面层和业务逻辑层之间，我们添加了 businesslogicservice.reserveblservice.ReserveBLSrevice 接口。业务逻辑层和数据层之间我们添加了 dataservice.reservedataservice.ReserveDataService 接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了 ReserveController，这样 ReserveController 会将对预定的业务逻辑梳理委托给 Reserve 对象。ReservePO 是作为预定记录的持久化对象被添加到设计模型中去的。而 ReservList 和 ReserveLineItem 的添加是 CommodityInfo 的容器类。ReserveLineItem 保有预定服务的数据，以及相应的计算费用的职责。而 ReserveList 封装了关于 ReserveLineItem 的数据集合的数据结构的秘密和计算总价的职责。ReserveInfo 和 MemberInfo 都是根据依赖倒置的原则，为了消除循环依赖而产生的接口。下图为 Reservebl 各个类的设计



Reservebl 模块的各个类的职责如下表所示。

模块	职责
LogicController	负责实现对应于登录界面所需要的服务
ReserveController	负责实现预定服务界面需要的服务
User	系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题
Reserves	预定服务的领域模型对象，拥有一次预定所持有的用户，场地，价格，预定记录等信息，可以帮助完成预定界面所需要的服务

4.1.3 模块内部的接口规范

下面两个表分别为 ReserveController 和 Reserve 的接口规范。

ReserveController 的接口规范

提供的服务（供接口）		
ReserveController.addMember	语法	Public ResultMessage addMember(long id)
	前置条件	已创建一个 Reserve 对象，并且输入符合输入规则
	后置条件	调用 Reserve 领域对象的 addMember 方法
ResverveController.addReserve	语法	PublicResultMessage addReserve (long id,long quantity)

	前置条件	已创建一个 Reserve 对象，并且输入符合输入规则
	后置条件	调用 Reserve 领域对象的 addReserve 方法
ResverveController.getTotal	语法	Public ResultMessage getTotal(long id,long quantity)
	前置条件	已创建一个 Reserve 对象，已添加预定用户和服务场地信息，并且输入符合输入规则
	后置条件	调用 Reserve 领域对象的 getTotal 方法
ResverveController.endReserve	语法	Public ResultMessage endReserve (long id,long quantity)
	前置条件	已创建一个 Reserve 对象
	后置条件	调用 Reserve 领域对象的 endReserve 方法
ResverveController.extraReserve	语法	Public ResultMessage extraReserve(long id,long quantity)
	前置条件	已创建一个 Reserve 对象
	后置条件	调用 Reserve 领域对象的 extraReserve 方法
需要的服务（需接口）		
服务名		服务
Reserve. addMember(long id)		加入一个预定用户对象
Reserve. addCommedity(long id,long quantity)		加入一个预定服务场地对象
Reserve. extraReserve (long id,long quantity)		加入额外商品信息
Reserve. getTotal(long id,long quantity)		计算预定总费用
Reserve. endReserve (long id,long quantity)		结束一次预定流程

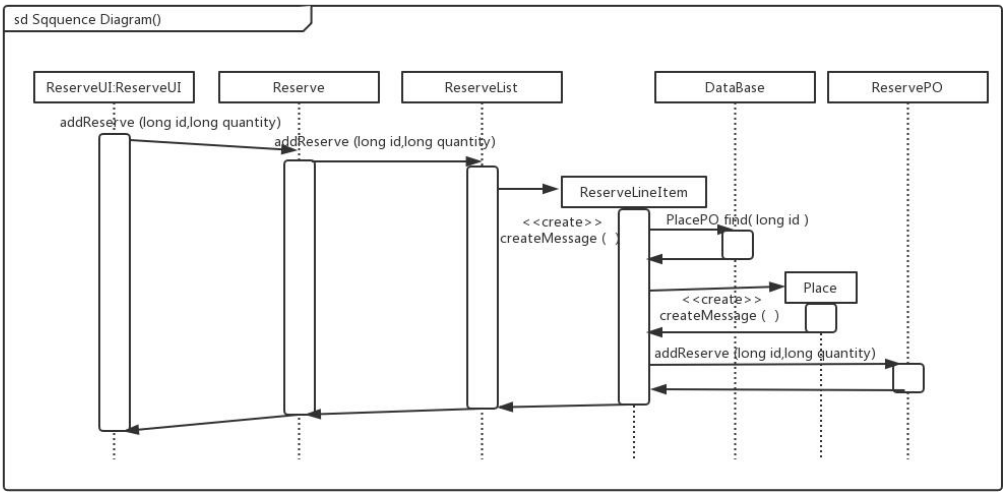
Reserve 的接口规范

提供的服务（供接口）		
Reserve.addMember	语法	Public ResultMessage addMember(long id)
	前置条件	启动一个服务场地预定回合
	后置条件	在一个服务场地预定回合中，增加预定的用户信息
Resverve.addReserve	语法	Public ResultMessage addReserve (long id,long quantity)
	前置条件	启动一个服务场地预定回合
	后置条件	在一个服务场地预定回合中，增加预定的场地信息
Resverve.getTotal	语法	Public ResultMessage getTotal(long id,long quantity)
	前置条件	已添加用户信息和预定服务场地信息
	后置条件	返回此次预定服务场地中需要支付的总额

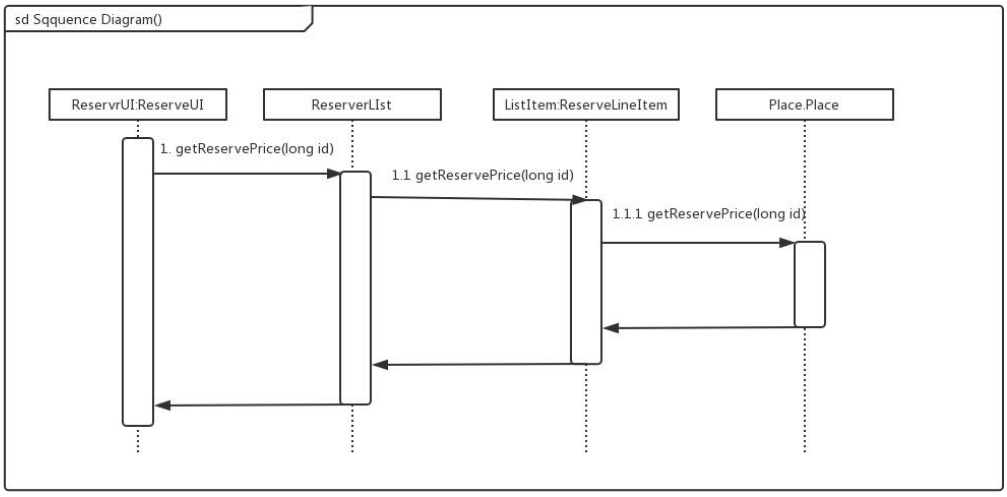
Resverve.endReserve	语法	Public ResultMessage endReserve (long id,long quantity)
	前置条件	已支付
	后置条件	结束此次服务场地预定回合，持久化更新涉及的领域对象的数据
Resverve.extraReserve	语法	Public ResultMessage extraReserve(long id,long quantity)
	前置条件	启动一个服务场地预定回合
	后置条件	增加额外预订服务信息，持久化更新涉及的领域对象的数据
需要的服务（需接口）		
服务名		服务
ReserveDataservice.find(long id)		根据 id 进行查找单一持久化对象
ReserveDataservice.insert(ReservePo po)		插入单一持久化对象
ReserveDataservice.delete(ReservePo po)		删除单一持久化对象
ReserveDataservice.update(ReservePo po)		更新单一持久化对象
ReserveDataservice.init(ReservePo po)		初始化单一持久化对象
ReserveDataservice.finish(ReservePo po)		结束持久化数据库的使用

4.1.4 数据层的动态模型

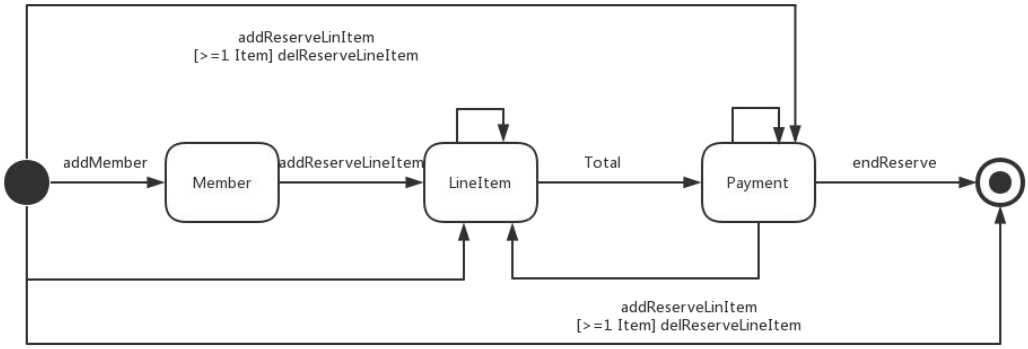
下图表示在智能车辆出行一体化系统中，单用户输入预定的服务场地信息后，预定业务逻辑处理的相关对象之间的协作。



下图为 Reserve 领域对象想要获得预定服务场地费用时候的顺序图。



下图所示的状态图描述了 Reserve 领域对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着 addMember 方法被 UI 调用，Reserve 进入 Member 状态；之后通过添加场地进入 LineItem 状态。UI 也可以不输入用户账号，直接添加服务场地进入 LineItem 状态。



4.1.5 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑有各自的控制器委托个不同的领域对象。

4.2 数据层的分解

数据层的开发包图参见软件体系结构设计文档。

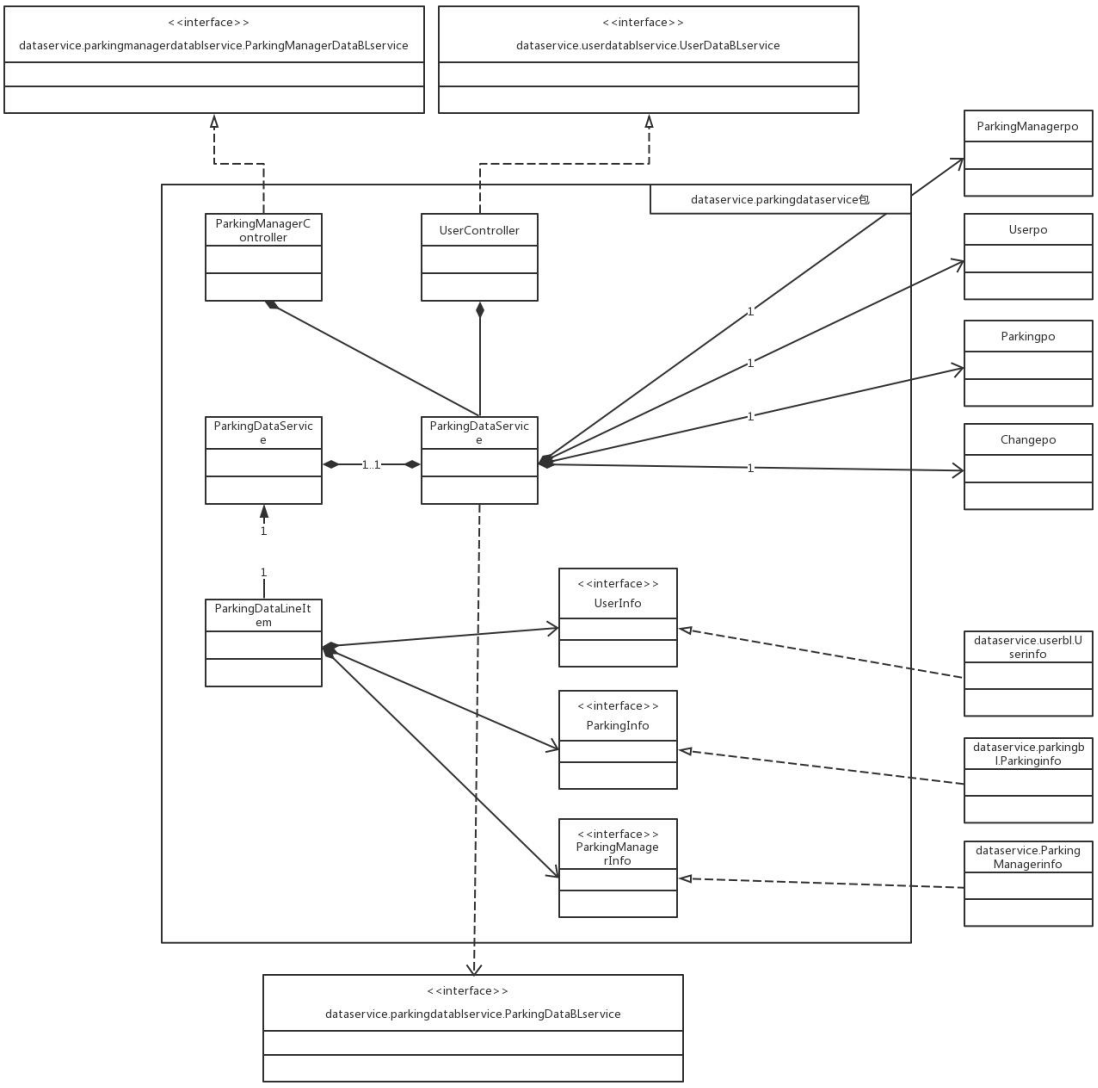
4.2.1 模块概述

ParkingLotDataServicebl 模块承担的需求参见需求规格说明文档功能需求说明和相关

非功能性需求说明. ParkingLotDataServicebl 模块的职责及接口参见软件体系结构设计文档。

4.2.2 整体结构

根据体系结构的设计，系统分为用户界面层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们会添加接口．业务逻辑层和数据层之间我们添加了dataservice.parkingdataservice.ParkingDataService 接口。为了隔离数据层职责和业务逻辑层 职责，我们增加了 UserController，这样 UserController 会将对车主用户管理信息进行委托给 ParkingDataService 对象。增加了 ParkingManagerController 和 ServiceManagerController，这样 ParkingManagerController 和 ServiceManagerController 会将对服务点和停车场管理员用户管理信息进行委托给 ParkingDataService 对象 ParkingDataLineItem 保有预定具体相关所有数据包含宿舍报修申请等各类内容。而 ParkingDataList 封装了关于 ParkingDataLineItem 的数据集合的数据结构的秘密。Usermemberinfo 、 ParkingManagermemberinfo 、 Servicemanagermemberinfo 、Reserveinfo、Announceinfo 都是根据依赖倒置的原则,为了消除循环依赖而产生的接口.下图为 ParkingDataServicebl 各个类的设计:



ParkingLotDataServicebl 模块的各个类的职责如下表所示:

模块	职责
UserController	负责实现对应于用户管理界面所需服务
ParkingManagerController	负责实现对应与停车场管理人员管理界面所需服务
ServiceManagerController	负责实现对应与服务点管理人员管理界面所需服务
ReserveController	负责服务预约处理界面所需服务
GuideController	负责服务目标地点导航界面所需服务
User	系统普通车主用户的领域模型对象，拥有普通车主用户数据的姓名和密码，可以解决普通车主用户登录问题
ParkingManager	系统停车场管理员的领域模型对象，拥有停车场管理员数据的姓名和密码，可以解决停车场管理员登录问题
ServiceManager	系统服务点管理员的领域模型对象，拥服务点管理员用户数据的姓名和密码，可以解决服务点管理员登录问题
UserDetail	系统普通车主用户的领域模型对象，拥有普通车主用户的所有具体信息，可以解决普通车主用户信息查询问题
ParkingManagerDetail	系统停车场管理员的领域模型对象，拥有停车场管理员用户的所有具体信息，可以解决停车场管理员用户信息查询问题
ServiceManagerDetail	系统服务点管理员的领域模型对象，拥有服务点管理员用户的所有具体信息，可以解决服务点管理员用户信息查询问题
Reserve	服务预约的领域模型对象，拥有一次服务预约所持有的车主用户信息，所选服务、预约时间等信息，可以帮助完成服务预约界面所需要的服务
Guide	目的地导航的领域模型对象，拥有一次目的地导航所持有的目的地经纬度，当前位置经纬度，导航用户编号等信息，可以帮助完成目的地导航界面所需要的服务

4.2.3 模块内部的接口规范

下面三个表分别为 UserController、ParkingManagerController 和 ServiceManager 的接口规范。

UserController 的接口规范

提供的服务（供接口）		
UserController.addUser	语法	Public ResultMessage addUser（long id）
	前置条件	信息输入符合输入规则
	后置条件	调用 User 领域对象的 addUser 方法
UserController.deleteUser	语法	Public ResultMessage deleteUser（long id）
	前置条件	该 User 对象已被创建
	后置条件	调用 User 领域对象的 deleteUser 方法
UserController.changeUser	语法	Public ResultMessage changeUser（long id）
	前置条件	该 Student 对象已被创建
	后置条件	调用 User 领域对象的 changeUser 方法
UserController.findUser	语法	Public ResultMessage findUser（long id）
	前置条件	该 User 对象已被创建

	后置条件	调用 User 领域对象的 findUser 方法
UserController.updateUser	语法	Public ResultMessage UpdateUser（long id）
	前置条件	该 User 对象已被创建
	后置条件	调用 User 领域对象的 updateUser 方法

ParkingManagerController 的接口规范

提供的服务（供接口）		
ParkingManagerController.addParkingManager	语法	Public ResultMessage addParkingManager（long id）
	前置条件	信息输入符合输入规则
	后置条件	调用 ParkingManager 领域对象的 addParkingManager 方法
ParkingManagerController.deleteParkingManager	语法	Public ResultMessage deleteParkingManager（long id）
	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 deleteParkingManager 方法
ParkingManagerController.changeParkingManager	语法	Public ResultMessage changeParkingManager（long id）
	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 changeParkingManager 方法
ParkingManagerController.findParkingManager	语法	Public ResultMessage findParkingManager（long id）
	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 findParkingManager 方法
ParkingManagerController.updateParkingManager	语法	Public ResultMessage updateParkingManager（long id）
	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 updateParkingManager 方法
ParkingManagerController.addAnnounceChange	语法	Public ResultMessage addAnnounceChange（long id）
	前置条件	信息输入符合输入规则
	后置条件	调用 ParkingManager 领域对象的 addAnnounceChange 方法
ParkingManagerController.deleteAnnounceChange	语法	Public ResultMessage deleteAnnounceChange（long id）
	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 deleteAnnounceChange 方法
ParkingManagerController.ch	语法	Public ResultMessage changeAnnounceChange（long id）

angeAnnounceChange	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 changeAnnounceChange 方法
ParkingManagerController.findAnnounceChange	语法	Public ResultMessage findAnnounceChange（long id）
	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 findAnnounceChange 方法
ParkingManagerController.updateAnnounceChange	语法	Public ResultMessage updateAnnounceChange（long id）
	前置条件	该 ParkingManager 对象已被创建
	后置条件	调用 ParkingManager 领域对象的 updateAnnounceChange 方法

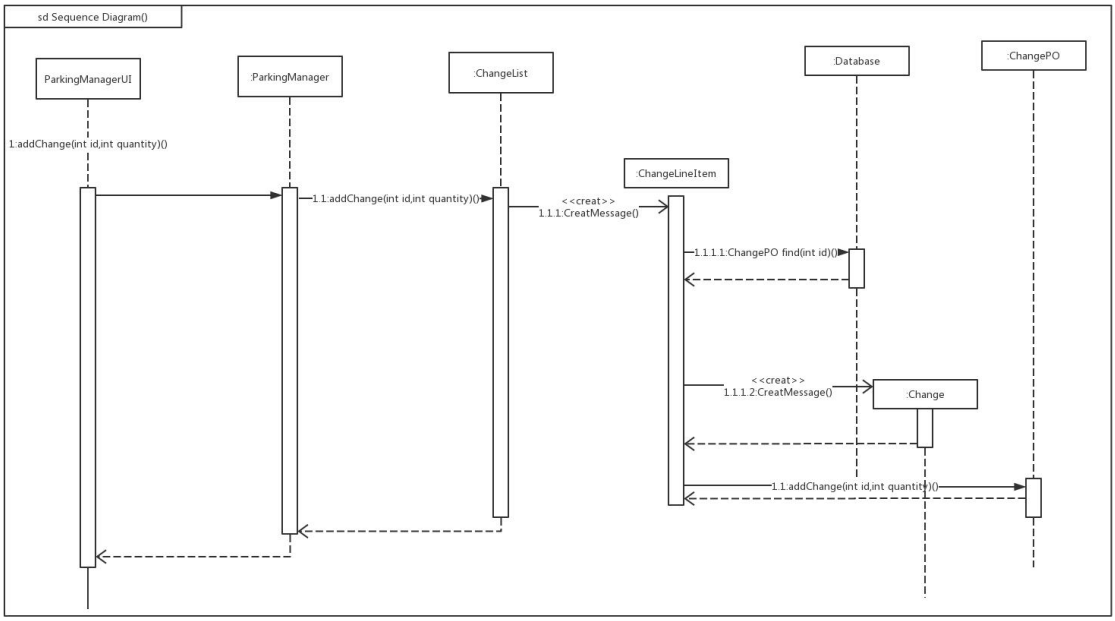
ServiceManagerController 的接口规范

提供的服务（供接口）		
ServiceManagerController.addServiceManager	语法	Public ResultMessage addParkingManager（long id）
	前置条件	信息输入符合输入规则
	后置条件	调用 ServiceManager 领域对象的 addServiceManager 方法
ServiceManagerController.deleteServiceManager	语法	Public ResultMessage deleteParkingManager（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 deleteServiceManager 方法
ServiceManagerController.changeServiceManager	语法	Public ResultMessage changeServiceManager（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 changeServiceManager 方法
ServiceManagerController.findServiceManager	语法	Public ResultMessage findServiceManager（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 findServiceManager 方法
ServiceManagerController.updateServiceManager	语法	Public ResultMessage updateServiceManager（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 updateServiceManager 方法
ServiceManagerController.add	语法	Public ResultMessage addAnnounceChange（long id）

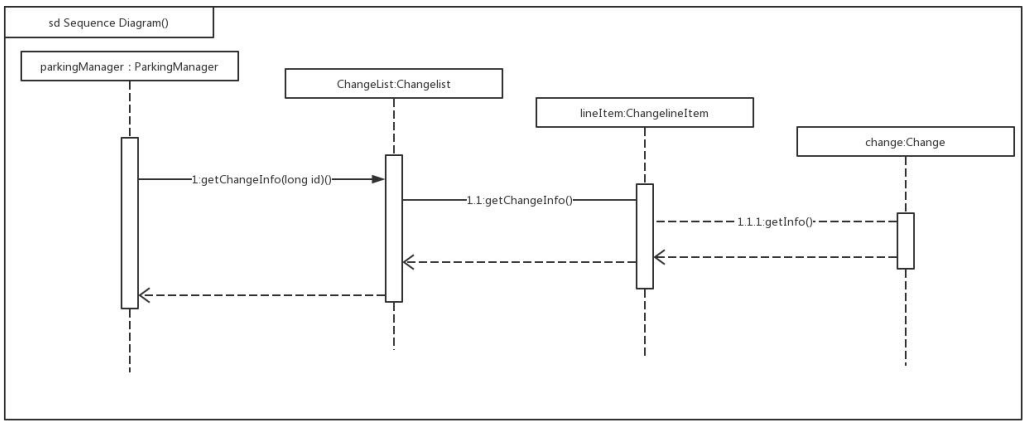
AnnounceChange	前置条件	信息输入符合输入规则
	后置条件	调用 ServiceManager 领域对象的 addAnnounceChange 方法
ServiceManagerController.deleteAnnounceChange	语法	Public ResultMessage deleteAnnounceChange（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 deleteAnnounceChange 方法
ServiceManagerController.changeAnnounceChange	语法	Public ResultMessage changeAnnounceChange（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 changeAnnounceChange 方法
ServiceManagerController.findAnnounceChange	语法	Public ResultMessage findAnnounceChange（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 findAnnounceChange 方法
ServiceManagerController.updateAnnounceChange	语法	Public ResultMessage updateAnnounceChange（long id）
	前置条件	该 ServiceManager 对象已被创建
	后置条件	调用 ServiceManager 领域对象的 updateAnnounceChange 方法

4.2.4 数据层的动态模型

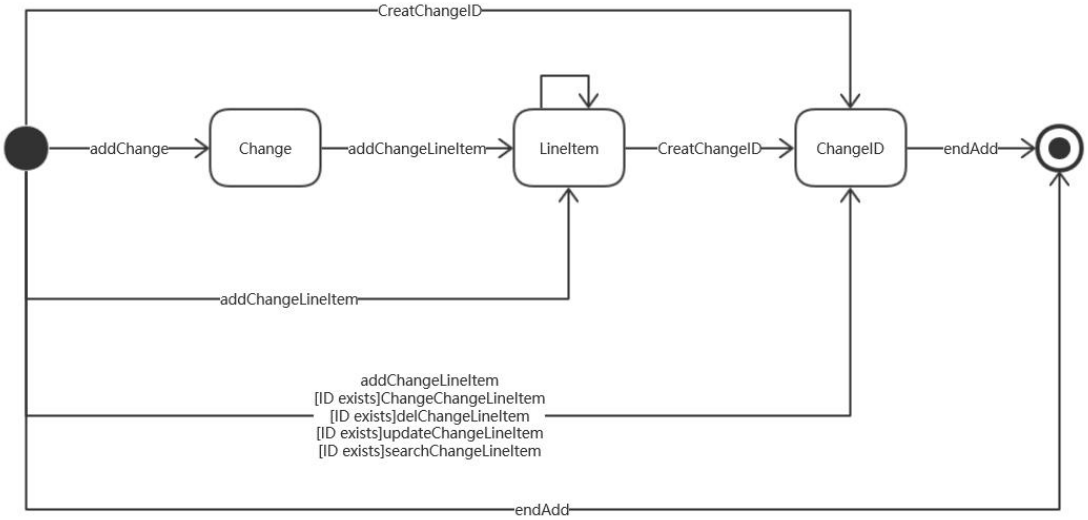
下图为智能出行车辆一体化服务管理系统中，停车场管理人员输入发布更改信息操作后，更改信息管理所产生的个对象间的协作关系：



下图为停车场管理人员 ParkingManager 领域对象想要进行查找更改信息时对应的顺序图关系：



下图所示的状态图描述了 ParkingManager 领域对象的生存期间的状态序列、引起转移的 事件，以及因状态转移而伴随的动作。随着 addChange 方法被 UI 调用，ParkingManager 进入 Change 状态；之后通过输入停车场管理人员账户密码进入 ChangeinfoLineItem 状态：

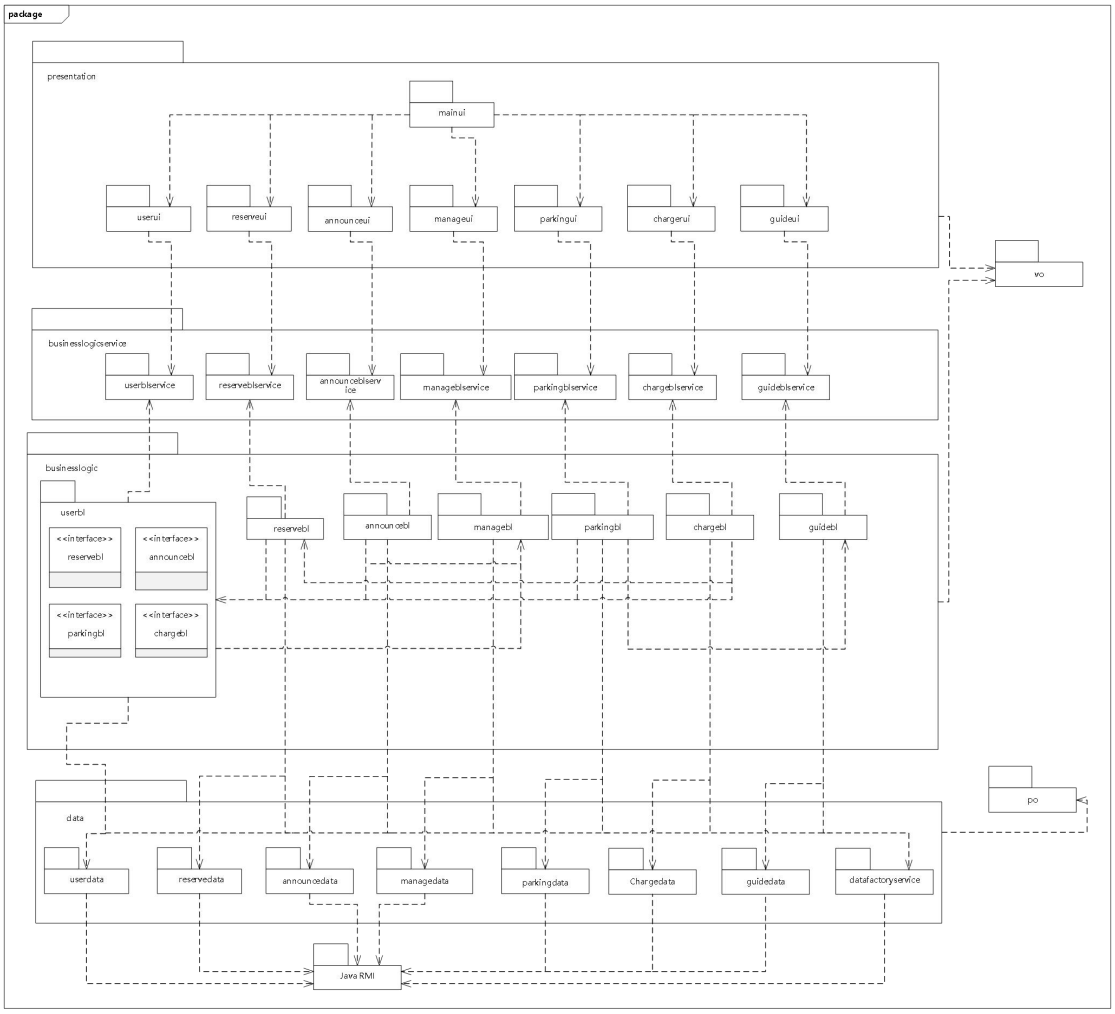


4.2.5 数据层的设计原理

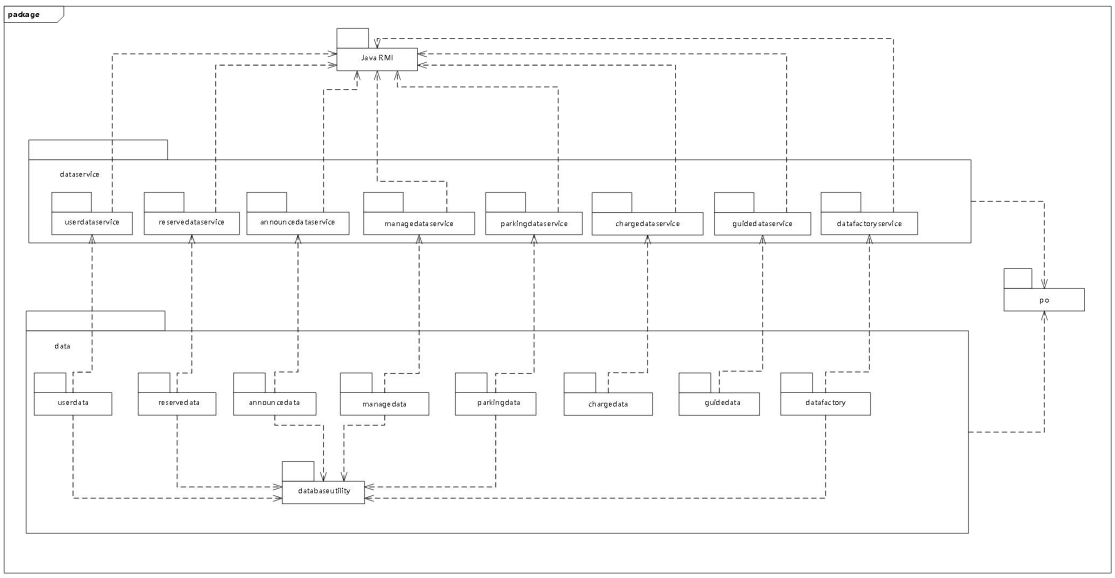
利用委托式控制风格，每个界面需要访问的业务逻辑有各自的控制器委托个不同的领域对象.

5 依赖视角

下面两图是客户端和服务器端各自的包之间的依赖关系。



客户端包图



服务器端包图