**快递物流系统**

**(Express Logistics System)**

**软件详细设计描述**



**学 院：南京大学软件学院**

**团 队：伪装者W小组**

**成 员：张词校 张家盛 王丽莉 魏彦淑**

**完成日期：2015年11月05日**

目录

[更新历史 2](#_Toc434506194)

[1.引言 3](#_Toc434506195)

[1.1编制目的 3](#_Toc434506196)

[1.2词汇表 3](#_Toc434506197)

[1.3参考资料 3](#_Toc434506198)

[2.产品概述 3](#_Toc434506199)

[3.体系结构设计概述 3](#_Toc434506200)

[4.结构视角 4](#_Toc434506201)

[4.1展示层的分解 4](#_Toc434506202)

[4.1.1模块职责 4](#_Toc434506203)

[4.1.2接口规范 4](#_Toc434506204)

[4.2业务逻辑层的分解 7](#_Toc434506205)

[4.2.1expressbl模块 7](#_Toc434506206)

[4.2.2businessbl模块 14](#_Toc434506207)

[4.2.3intermediatebl模块 27](#_Toc434506208)

[4.2.4receiptbl模块 41](#_Toc434506209)

[4.2.5repertorybl模块 45](#_Toc434506210)

[4.2.6financebl模块 53](#_Toc434506211)

[4.2.7managebl模块 68](#_Toc434506212)

[4.2.8userbl模块 82](#_Toc434506213)

[4.3数据层的分解 85](#_Toc434506214)

[4.3.1模块职责 85](#_Toc434506215)

[4.3.2接口规范 86](#_Toc434506216)

[5.依赖视角 99](#_Toc434506217)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 全体成员 | 2015.11.04 | 完成初稿 | V0.1 |
| 魏彦淑、王丽莉  张家盛 | 2015.11.05  2015.11.12 | 对初稿进行汇总完善  新增目录，更新历史，依赖关系 | V1.0  V2.0 |

# 1.引言

## 1.1编制目的

本报告详细完成对进销存系统的概要设计，达到指导详细设计和开发的目的，同时实现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而编写，是了解系统的导航。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统（Express Logistics System） | 无 |
| BL | 业务逻辑（Business　Logic） | 无 |

## 1.3参考资料

[IEEE1471-2000]

快递物流系统用例文档和快递物流系统需求规格说明。

# 2.产品概述

参考快递物流系统用例文档和快递物流系统需求规格说明对产品的概括描述。

# 3.体系结构设计概述

参考快递物流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展示层的分解

### 4.1.1模块职责

如表1所示为用户界面层模块的职责。

**表1 用户界面层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面Frame，负责界面的显示和跳转 |
| ExpressFrame | 快递员部分界面Frame，负责快递员部分界面的显示与跳转 |
| BusinessFrame | 营业厅部分界面Frame，负责营业厅部分界面的显示和跳转 |
| IntermediateFrame | 中转中心部分界面Frame，负责中转中心部分界面的显示和跳转 |
| ReceiptFrame | 单据部分界面Frame，负责单据部分界面的显示 |
| FinanceFrame | 财务部分界面Frame，负责财务部分界面的显示和跳转 |
| RepertoryFrame | 仓库部分界面Frame，负责仓库部分界面的显示与跳转 |
| ManageFrame | 总经理部分界面Frame，负责总经理部分界面的显示和跳转 |
| UserFrame | 用户部分界面Frame，负责用户登录界面和管理员增删改查用户界面的显示和跳转 |

### 4.1.2接口规范

用户界面层模块的接口规范如表2所示。

**表2 用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | Frameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示Frame |
| ExpressFrame | 语法 | ExpressFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示快递员部分界面的Frame以及订单新建，订单完成，订单查询和订单确认收费的panel |
| BusinessFrame | 语法 | BusinessFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示营业厅部分界面的Frame以及装车管理，车辆和司机的信息管理，派件和收费汇总的panel |
| IntermediateFrame | 语法 | IntermediateFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示中转中心部分界面的Frame以及飞机，火车，汽车装运管理和中转接收的panel |
| ReceiptFrame | 语法 | ReceiptFrameFrameini（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示单据部分界面的Frame以及各种单据查增删改的panel |
| FinanceFrame | 语法 | FinanceFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示财务界面的Frame以及表单新建，报表查看，期初建账和账户管理的panel |
| RepertoryFrame | 语法 | RepertoryFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示仓库界面的Frame以及商品出、入库管理,库存初始化，库存报警，库存查看和库存盘点的panel |
| ManageFrame | 语法 | ManageFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示总经理界面的Frame以及机构管理，人员管理，基础数据设置，财务报表查看的panel |
| UserFrame | 语法 | UserLogInFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示用户登录界面的Frame以及跳转 |
| UserFrame | 语法 | UserManageFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示管理员界面的Frame以及账号管理的panel |

用户界面层需要的服务接口如表7所示。

**表7 用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| **服务名** | **服务** |
| businesslogicservice.userblservice.UserBLService | 用户信息的增删改查的业务逻辑接口和登录界面的业务逻辑接口 |
| businesslogicservice.businessblservice.DriverManagerBLService  businesslogicservice.businessblservice.VehicleManagerBLService  businesslogicservice.businessblservice.EnVehicleBLService  businesslogicservice.businessblservice.GatheringBLService  businesslogicservice.businessblservice.DistributeOrderBLService  businesslogicservice.businessblservice.AcceptCargoBLService | 营业厅司机信息的增删改查的业务逻辑接口  营业厅车辆信息的增删改查的业务逻辑接口  营业厅车辆装车安排及的业务逻辑接口  营业厅每日收款单列表提交的业务逻辑接口  营业厅派件的业务逻辑接口  营业厅接收货物的业务逻辑接口 |
| businesslogicservice.expressblservice.LogisticQueryBLService  businesslogicservice.expressblservice.AddOrderBLService  businesslogicservice.expressblservice.ChargeCollectionBLService  businesslogicservice.expressblservice.ReceiptOrderBLService | 快递员查询订单的业务逻辑接口  快递员输入订单的业务逻辑接口  快递员进行收费信息汇总的业务逻辑接口  快递员收件信息输入的业务逻辑接口 |
| businesslogicservice.financeblservice.CollectionReceiptBLService  businesslogicservice.financeblservice.PaymentReceiptBLService  businesslogicservice.financeblservice.CostIncomeReceiptBLService  businesslogicservice.financeblservice.BusinessstatementReceiptBLService  businesslogicservice.financeblservice.InitialStockBLService  businesslogicservice.financeblservice.AccountBLService | 结算管理的业务逻辑接口  新建付款单的业务逻辑接口  新建成本收益表的业务逻辑接口  查看经营情况表的业务逻辑接口  期初建账的业务逻辑接口  账户的增删改查的逻辑接口 |
| businesslogicservice.intermediateblservice.EnplaningBLService  businesslogicservice.intermediateblservice.EntrainingBLService  businesslogicservice.intermediateblservice.EntruckingBLService  businesslogicservice.intermediateblservice.TransferingBLService | 飞机装运管理的业务逻辑接口  火车装运管理的业务逻辑接口  汽车装运管理的业务逻辑接口  中转接收的业务逻辑接口 |
| businesslogicservice.repertoryblservice.RepertoryBLService  businesslogicservice.repertoryblservice.GoodsBLService  businesslogicservice.repertoryblservice.EnterRepertoryReceiptBLService  businesslogicservice.repertoryblservice.LeaveRepertoryReceiptBLService | 出库、入库、库存查看、库存盘点、库存信息初始化的业务逻辑接口  对进出仓库涉及到的货物信息增删改查的业务逻辑接口  入库单信息的增删改查的业务逻辑接口  出库单信息的增删改查的业务逻辑接口 |
| businesslogicservice.manageblservice.OrganizationBLService  businesslogicservice.manageblservice.ReviewReceiptBLService  businesslogicservice.manageblservice.PerWageBLService  businesslogicservice.manageblservice.BasicSalaryBLService  businesslogicservice.manageblservice.CityDistanceBLService  businesslogicservice.manageblservice.CostBLService | 机构信息的增删改查的业务逻辑接口  单据查看并审批的业务逻辑接口  每次工资信息的增删改查的业务逻辑接口  基础月薪信息的增删改查的业务逻辑接口  城市距离信息的增删改查的业务逻辑接口  运费系数信息的增删改查的业务逻辑接口 |

## 4.2业务逻辑层的分解

### 4.2.1expressbl模块

#### 4.1.1.1模块概述

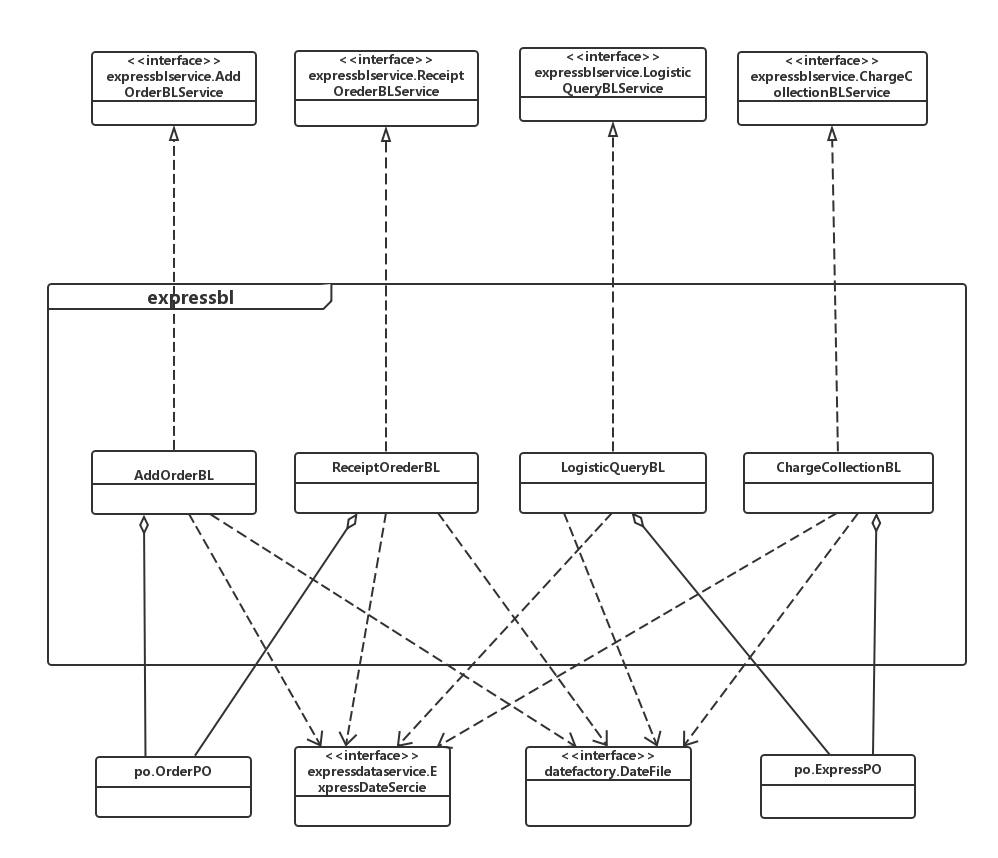
expressbl模块承担的需求参见ELS需求规格文档功能需求及相关非功能需求。

expressbl模块的职责及接口参见ELS体系结构设计文档。

#### 4.1.1.2整体结构

expressbl模块实现了展示层与业务逻辑层之间的expressblservice接口以及业务逻辑层和数据层之间的expressdataservice接口。提供了查询订单的calculate和addOrder方法,收件信息输入的getOrderInfo和receiptOrder方法,收费信息汇总的getChargeInfo和chargeCollection方法以及查询订单的query方法。查询订单，收件信息输入，收费信息汇总和查询订单用例分别对应类LogisticQueryBL,ReceiptOrderBL,ChargeCollectionBL和AddOrderBL类，

expressbl模块的设计如图1-1所示



**图1-1 expressbl模块各个类的设计**

expressbl各个类的职责如表1-1所示

**表1-1 expressbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| LogisticQueryBL | 负责实现快递员进行物流查询的服务 |
| ReceiptOrderBL | 负责实现快递员进行收件信息输入的服务 |
| ChargeCollectionBL | 负责实现快递员进行收费信息汇总的服务 |
| AddOrderBL | 负责实现快递员新增订单的服务 |

#### 4.1.1.3模块内部类的接口规范

LogisticQueryBL,ReceiptOrderBL,ChargeCollectionBL和AddOrderBL的接口规范如表1-2，表1-3，表1-4和表1-5所示

**表1-2 LogisticQueryBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Express.LogisticQuery.getOrderInfo(String orderNum) | 语法 | public OrderVO getOrderInfo(String id) | |
|  | 前置条件 | 快递员经过验证有查询订单的权利 | |
| 后置条件 | 系统查找订单是否存在，如果找到就返回订单信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDataService.find(String orderID) | | | 查找订单信息 |

**表1-3 ReceiptOrderBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Express.ReceiptOrder.getOrderInfo | 语法 | public OrderVO getOrderInfo(long orderNum) | |
| 前置条件 | 快递员经过验证有确认订单的权利，并要求查看订单信息 | |
| 后置条件 | 系统根据输入的订单号返回订单信息 | |
| Express.ReceiptOrde.receiptOrder | 语法 | public boolean receiptOrder(OrderVO vo) | |
| 前置条件 | 快递员经过验证有确认订单的权利，并要求确认订单 | |
| 后置条件 | 系统根据输入的订单接收信息更新订单，并返回更新结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDataService.changeState(OrderState state, String orderID) | | | 改变订单状态 |
| ExpressDataService.addHistory(String history,String organizationID,String orderID) | | | 记录订单的历史转运流程 |
| ExpressDataService.receiptOrder(String organizationID,String userID, OrderPO po) | | | 完成订单 |

**表1-4 ChargeCollectionBL的接口规范**

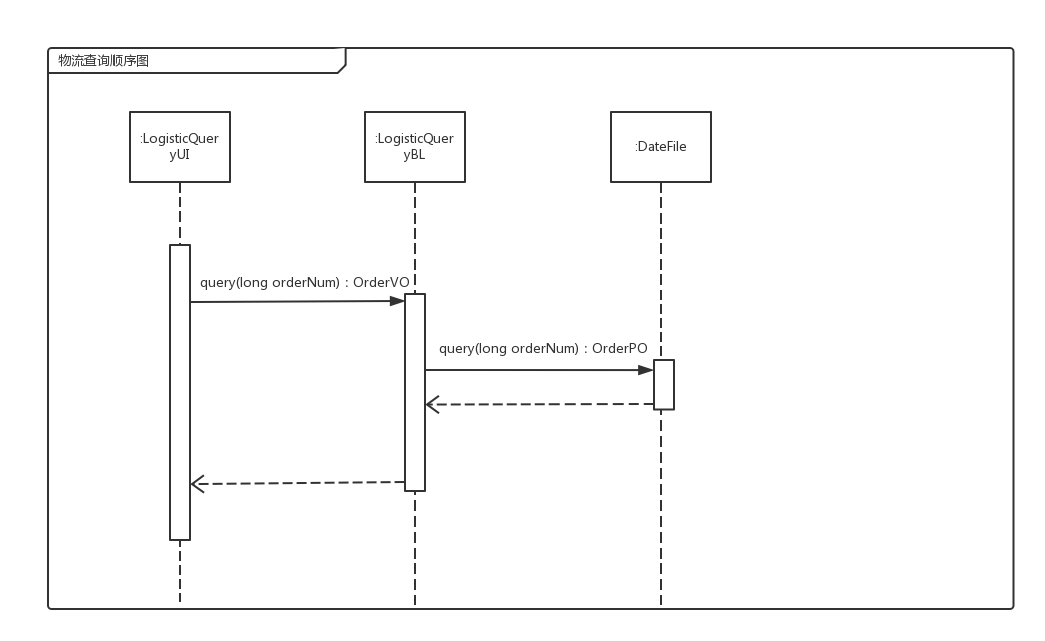
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| AddOrder.chargeCollection() | 语法 | public boolean chargeCollection(String chargeInfo) | |
| 前置条件 | 快递员经过验证有汇总收费的权利，并要求查看当前收费情况 | |
| 后置条件 | 系统更新快递员信息，并更具更新返回结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressData.updateChargeCollection(String organizationID,String userID,ArrayList<String> chargeCollection) | | | 更新快递员的收费记录 |

**表1-5 AddOrderBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Express.AddOrder.addOrder | 语法 | public boolean addOrder(OrderVO vo) | |
| 前置条件 | 快递员经过验证有添加订单的权利，输入了有效的待添加的订单信息 | |
| 后置条件 | 系统更新对应的订单信息，并根据更新结果返回 | |
| Express.AddOrder.calculate | 语法 | public void calculateCost(OrderVO vo | |
| 前置条件 | 快递员经过验证有新增订单的权利，并要求计算订单信息和估计送达时间 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| CityDistanceDataService.findCityDistanceByBoth(String city1,String city2) | | | 获得两城市间距离 |
| OrganizationDataService.getBelongingPlaces(String city) | | | 获得的市级下的地区名 |
| ExpressData.updateChargeCollection(String organizationID,String userID,ArrayList<String> chargeCollection) | | | 更新快递员的收费记录 |
| ExpressDataService.addOrder(OrderPO po, String organizationID) | | | 新增订单 |
| ExpressDataService.addSubmitOrder(String organizationID, String userID, String ID) | | | 更新快递员的新建订单列表 |

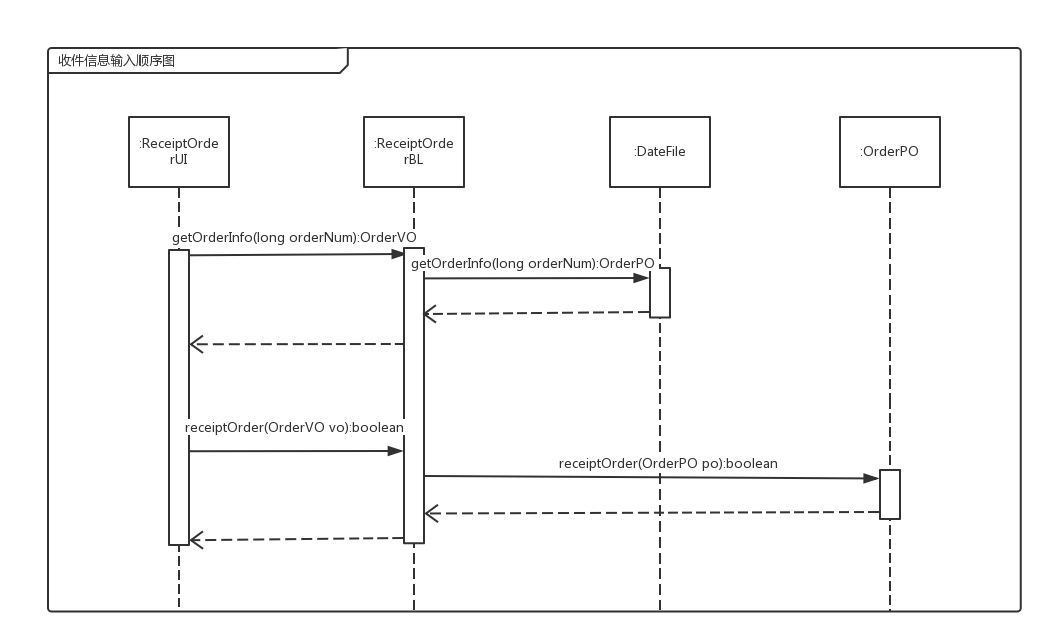
#### 4.1.1.4业务逻辑层的动态模型

图1-2为LogisticQueryBL领域对象想要新增订单时的顺序图



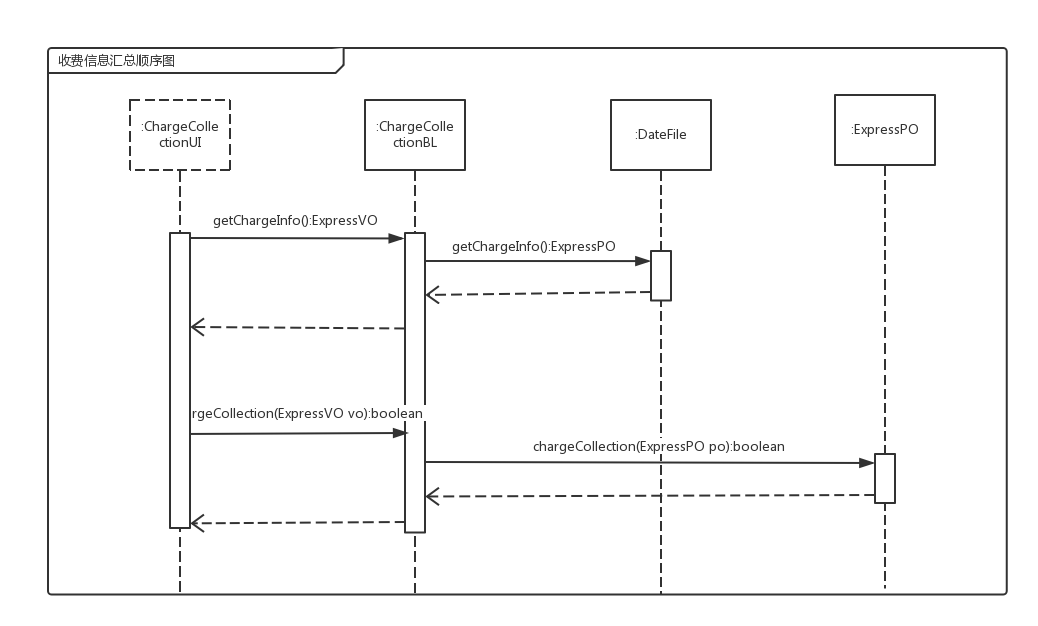
**图1-2物流查询的顺序图**

图1-3为ReceiptOrderBL领域对象想要进行收件信息输入时的顺序图



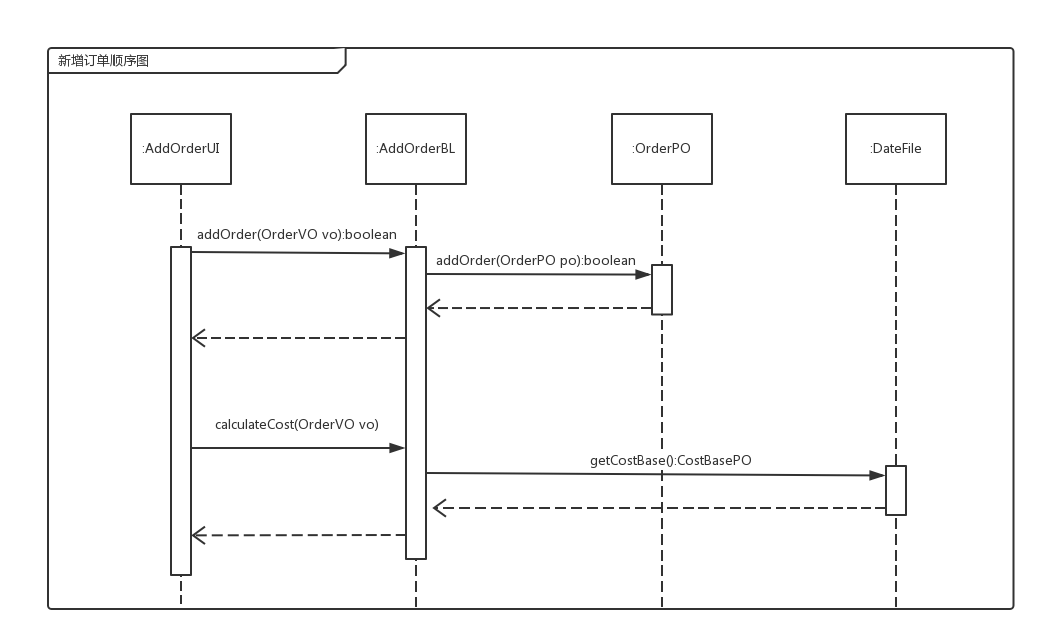
**图1-3 收件信息输入的顺序图**

图1-4为ChargeCollectionBL领域对象想要进行收费信息汇总时的顺序图



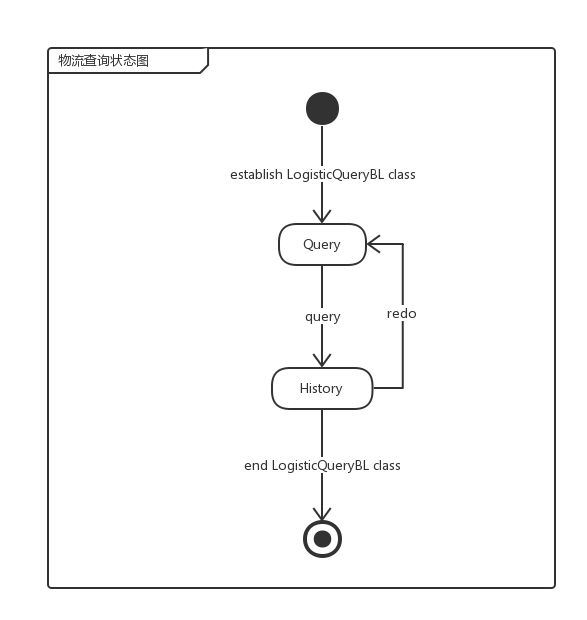
**图1-4 收费信息汇总的顺序图**

图1-5为AddOrderBL领域对象想要新增订单时的顺序图



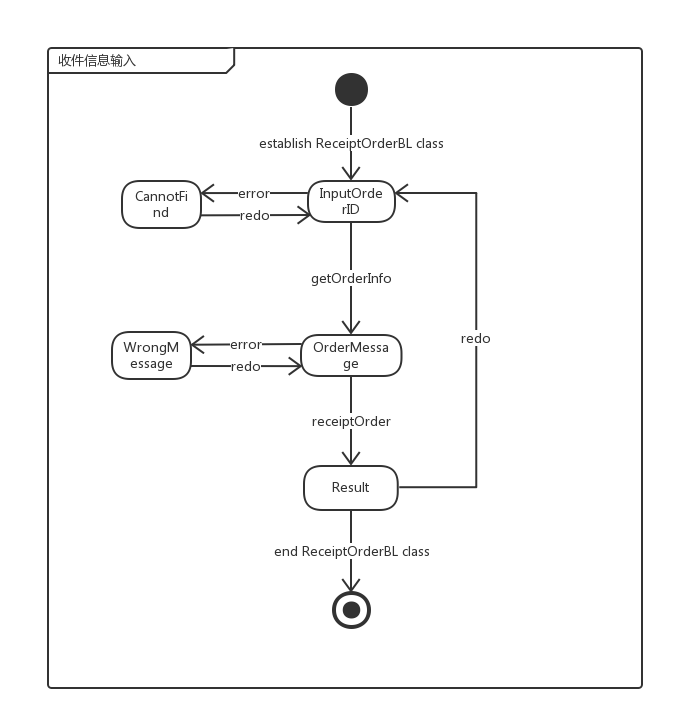
**图1-5新增订单的顺序图**

如图1-6所示，AddOrderBL对象状态图描述了AddOrderBL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establishAddOrderBL方法被 UI调用，AddOrderBL根据不同的指令进入不同状态。



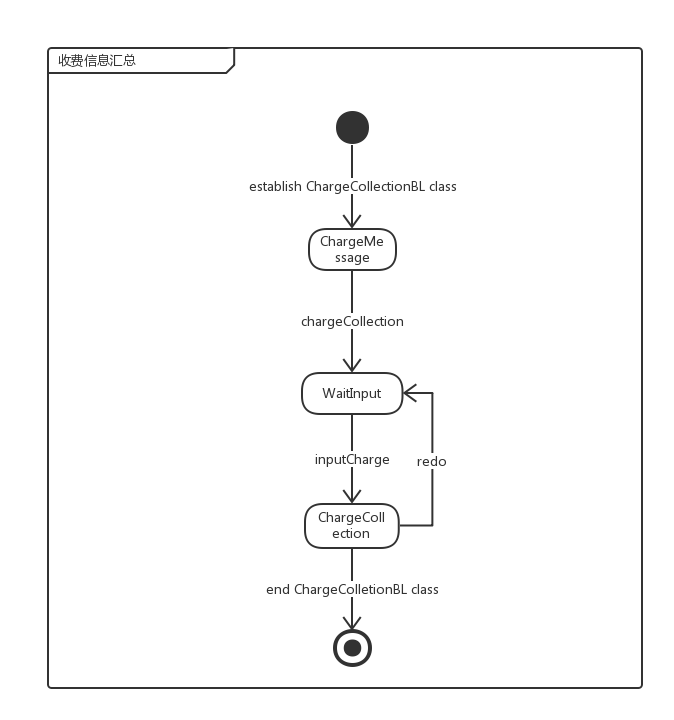
**图1-6 物流查询的状态图**

如图1-7所示，ReceiptOrderBL对象状态图描述了ReceiptOrderBL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establishReceiptOrderBL方法被 UI调用，ReceiptOrderBL根据不同的指令进入不同状态。



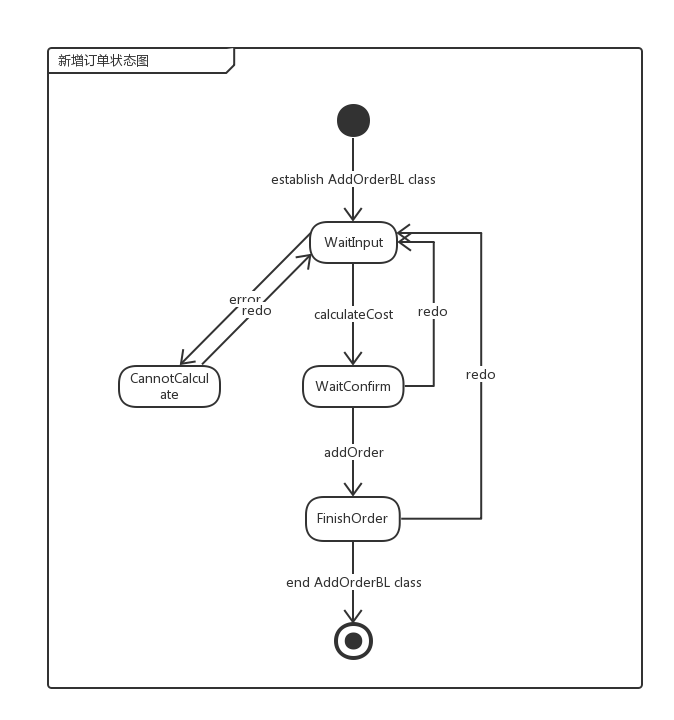
**图1-7 收件信息输入的状态图**

如图1-8所示，ChargeCollectionBL对象状态图描述了ChargeCollectionBL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establishChargeCollectionBL方法被 UI调用，ChargeCollectionBL根据不同的指令进入不同状态。



**图1-8 收费信息输入的状态图**

如图1-9所示，AddOrderBL对象状态图描述了AddOrderBL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establishAddOrderBL方法被 UI调用，AddOrderBL根据不同的指令进入不同状态。



**图1-9 新增订单的状态图**

### 4.2.2businessbl模块

#### 4.2.2.1模块概述

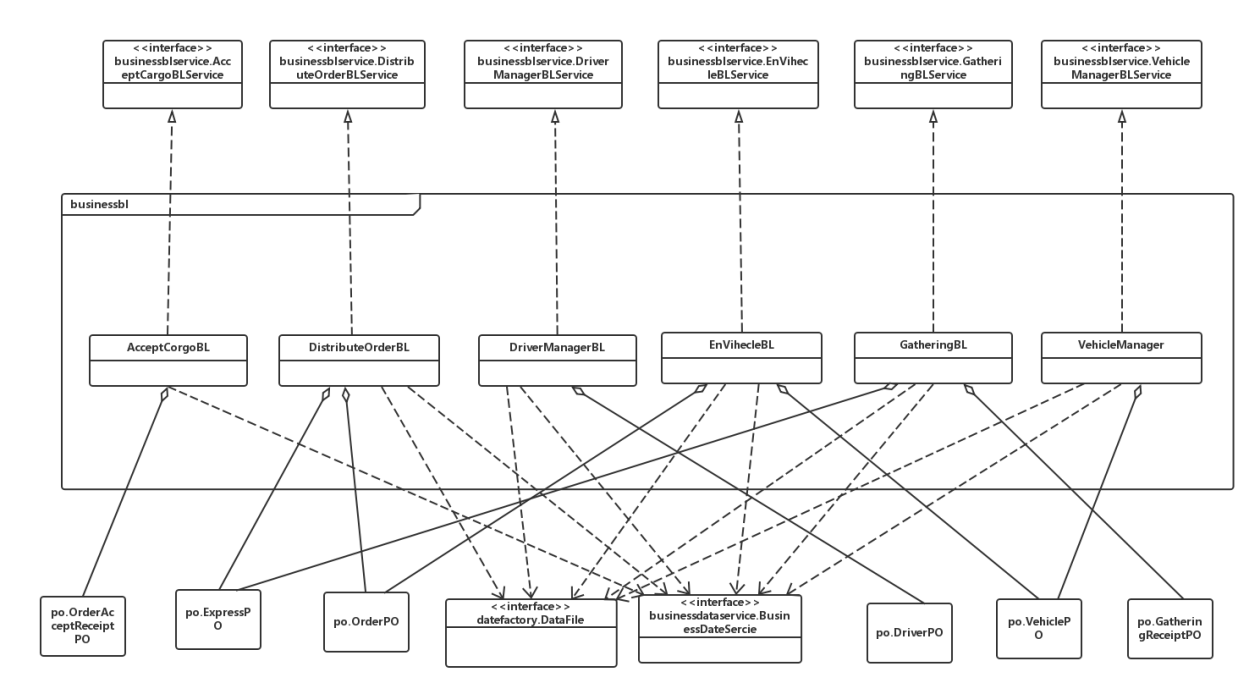
businessbl模块承担的需求参见ELS需求规格文档功能需求及相关非功能需求。

businessbl模块的职责及接口参见ELS体系结构设计文档。

#### 4.2.2.2整体结构

businessbl模块实现了展示层和业务逻辑层之间的businessblservice接口以及业务逻辑层和数据层之间的businessdataservice接口。提供了接收货物的acceptCargo方法，装车管理的autoTruckLoading方法，司机信息管理的addDriver，deleteDriver，modifyDriver方法，车辆信息管理的addVehicle，deleteVehicle，modifyVehicle方法，派件的distributeOrder方法以及收款汇总的gathering方法，接收货物，装车管理，司机信息管理，车辆信息管理，派件，收款汇总用例分别对应类AcceptCargoBL，EnVehicleBL，DriverManagerBL，VehicleManagerBL，DistributeOrderBL，GatheringBL类。

businessbl模块的设计如图2-1所示



**图2-1 businessbl模块各个类的设计**

businessbl各个类的职责如表2-1所示

**表2-1 expressbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| AcceptCargoBL | 负责实现营业厅业务员接收货物的服务 |
| EnVehicleBL | 负责实现营业厅业务员的进行装车管理服务 |
| DriverManagerBL | 负责实现营业厅业务员的司机信息管理服务 |
| VehicleManagerBL | 负责实现营业厅业务员的车辆信息管理服务 |
| DistributeOrderBL | 负责实现营业厅业务员派件的服务 |
| GatheringBL | 负责实现营业厅业务员的收款汇总服务 |

#### 4.2.2.3businessbl模块内部类的接口规范

AcceptCargoBL，EnVehicleBL，DriverManagerBL，VehicleManagerBL，DistributeOrderBL，GatheringBL的接口规范如表2-2，表2-3，表2-4，表2-5，表2-6和表2-7所示

**表2-2 AcceptCargoBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Business.AcceptCargo.acceptCargo | 语法 | public boolean acceptCargo(String vehicleID,OrderAcceptReceiptVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行接收货物的权利，并输入了所有货物的订单号 | |
| 后置条件 | 系统生成新的营业厅到达单并更新营业厅到达单列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDataService.changeState(OrderState state, String orderID) | | | | 改变订单状态 |
| ExpressDataService.addHistory(String history,String organizationID,String orderID) | | | | 记录订单的历史转运流程 |
| ExpressDataService.addDistributingOrder(ArrayList<String> orderIDs,String organizationID) | | | 更新快递员的派件信息 |
| BusinessDataService.getNumOfOrderAcceptReceipt(String organizationID); | | | 获得当日本的到达单数 |
| BusinessDataService.addReceipt(String oID, OrderAcceptReceiptVO vo) | | | 增加到达单 |

**表2-3 EnVehicleBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Business.EnVehicle.autoTruckLoading | 语法 | public ArrayList<String> autoTruckLoading() | |
| 前置条件 | 营业厅业务员经过验证有进行装车管理的权利，并要求进行装车管理任务 | |
| 后置条件 | 系统自动根据已有信息生成装车单，并更新到装车单列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDataService.getOrderInfoByTime(String organiztionID, String time) | | | 根据时间获取该营业厅新增的订单 |
| ExpressDataService.changeState(OrderState state, String orderID) | | | | 改变订单状态 |
| ExpressDataService.addHistory(String history,String organizationID,String orderID) | | | | 记录订单的历史转运流程 |
| BusinessDataService.getVehicleInfos(String organizationID); | | | 获得该营业厅的车辆信息 |
| BusinessDataService.addDriverTime(String organizationID,String driverID) | | | 增加司机运货次数 |
| BusinessDataService.addEnVehicleReceipt(String organizationID,EnVehicleReceiptPO po) | | | 新增营业厅装车单 |

**表2-4 DriverManagerBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Business.DistributeOrder.distributeOrder | 语法 | public ArrayList<String> distributeOrder() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求进行分配订单任务 | |
| 后置条件 | 系统根据已有信息生成装车单，并返回分配结果 | |
| Business.DriverManager.addVehicle | 语法 | public boolean addVehicle(Vehicle vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了正确的车辆信息 | |
| 后置条件 | 系统根据输入的车辆信息更新车辆列表，并返回更新结果 | |
| Business.DriverManager.getDriverInfo | 语法 | Public ArrayList<VehicleVO>getVehicleInfo(String ID) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入正确的车辆编号 | |
| 后置条件 | 系统返回已有车辆的信息 | |
| Business.DriverManager.deleteDriver | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求删除指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| Business.DriverManager.modifyDriver | 语法 | public boolean modifyVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求修改 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BusinessDataService.getDriverInfo() | | | 得到ID对应的司机信息 |
| BusinessDataService.addDriver(DriverPO po) | | | 增加司机信息 |
| BusinessDataService.deleteDriver(DriverPO po) | | | 删除司机信息 |
| BusinessDataService.modifyDriver(DriverPO po) | | | 修改司机信息 |

**表2-5 VehicleManager的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Business.VehicleManager.modifyVehicle | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求改正指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| Business.VehicleManager.addVehicle | 语法 | public boolean addVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了正确的车辆信息 | |
| 后置条件 | 系统根据输入的车辆信息更新车辆列表，并返回更新结果 | |
| Business.VehicleManager.getVehicleInfos | 语法 | public ArrayList<VehicleVO> getVehicleInfos() | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利 | |
| 后置条件 | 系统返回已有车辆的信息 | |
| Business.VehicleManager.deleteVehicle | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求删除指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BusinessDataService.addVehicle(VehiclePO po) | | | 增加车辆信息 |
| BusinessDataService.deleteVehicle(VehiclePO po) | | | 删除车辆信息 |
| BusinessDataService.modifyVehicle(VehiclePO po) | | | 修改车辆信息 |
| BusinessDataService.getVehicleInfos(String organizationID); | | | 获得该营业厅的车辆信息 |

**表2-6 DistributeOrderBL的接口规范**

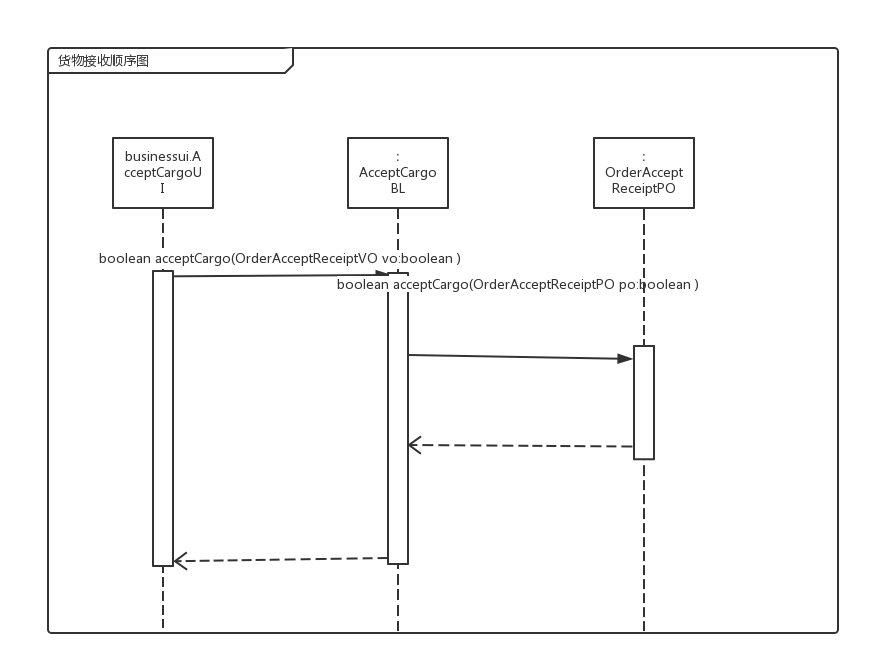
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Business.DistributeOrder.distributeOrder | 语法 | public ArrayList<String> distributeOrder() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求进行分配订单任务 | |
| 后置条件 | 系统根据已有信息生成装车单，并返回分配结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDataService.addDistributingOrder(ArrayList<String> orderIDs,String organizationID) | | | 更新快递员的派件信息 |
| ExpressDataService.changeState(OrderState state, String orderID) | | | | 改变订单状态 |
| ExpressDataService.addHistory(String history,String organizationID,String orderID) | | | | 记录订单的历史转运流程 |
| ExpressDataService.addPendingOrder(String organizationID, String userID, String orderID) | | | 增加快递员的派件列表 |
| ExpressDataService.getOrderInfoByTime(String organiztionID, String time) | | | 根据时间获取该营业厅新增的订单 |

**表2-7 GatheringBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Business.Gathering.getChargeInfo | 语法 | public ArrayList<ExpressVO>getChargeInfo() | |
| 前置条件 | 营业厅业务员经过验证有进行收款汇总的权利，并要求查看所有营业厅业务员的收款与订单信息 | |
| 后置条件 | 系统返回该营业厅所有快递员的信息（ID与收款） | |
| Business.Gathering.gathering | 语法 | public double gathering() | |
| 前置条件 | 营业厅业务员经过验证有进行收款汇总的权利，并要求进行收款汇总任务 | |
| 后置条件 | 系统汇总收款并返回结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ExpressDataService.deleteChargeInfos(String organizationID) | | | 删除本营业厅快递员的收费信息 |
| ExpressDataService.updateChargeCollection(String organizationID, i.getID(), ArrayList<String> chargeCollection) | | | 更行本营业厅快递员的收费信息 |
| BusinessDataService.addGatheringReceipt(String organizationID, GatheringReceiptPO po) | | | 新增入款单 |
| ExpressDataService.getExpressInfos(String oID) | | | 获取该营业厅的快递员信息 |

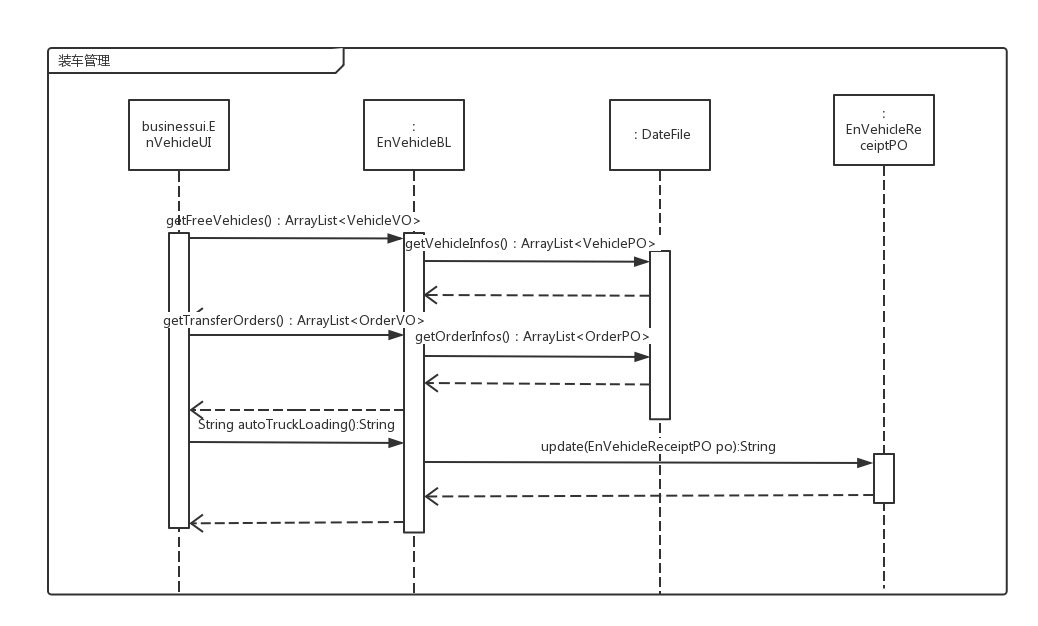
#### 4.2.2.4业务逻辑层的动态模型

图2-2为 AcceptCargoBL领域对象想要进行货物接收时的顺序图



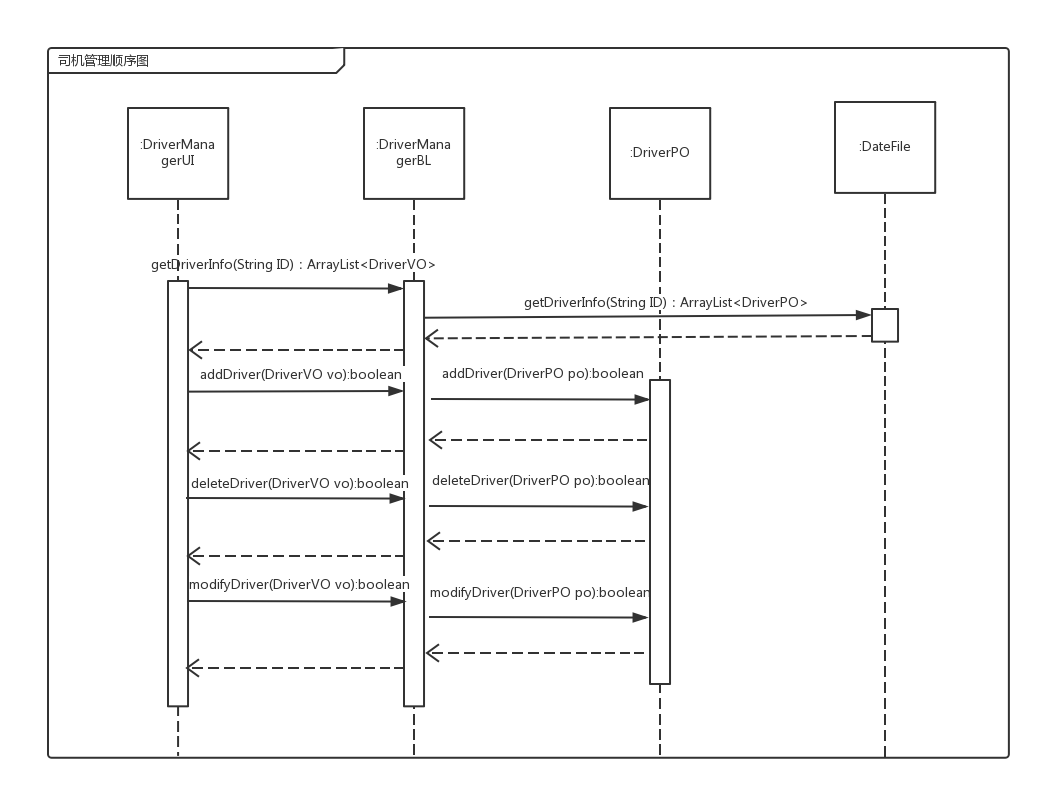
**图2-2 货物接收的顺序图**

图2-3为 EnVehicleBL领域对象想要进行装车管理时的顺序图



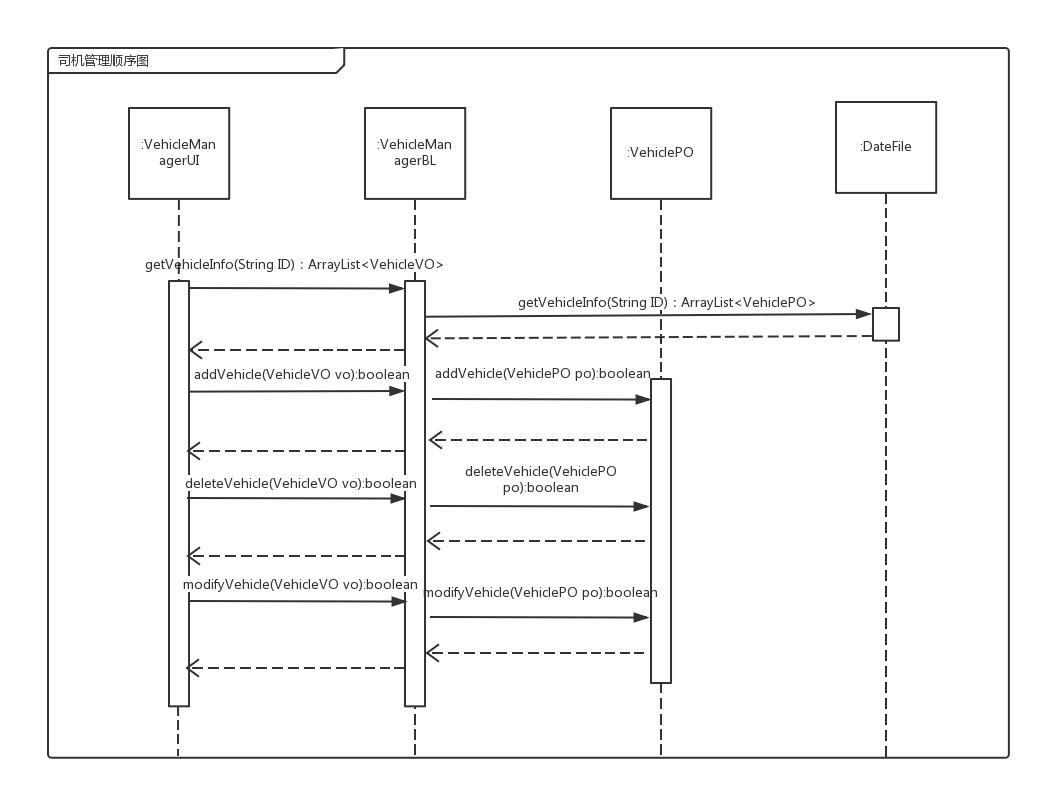
**图2-3 装车管理的顺序图**

图2-4为 DriverManagerBL领域对象想要进行司机信息管理时的顺序图



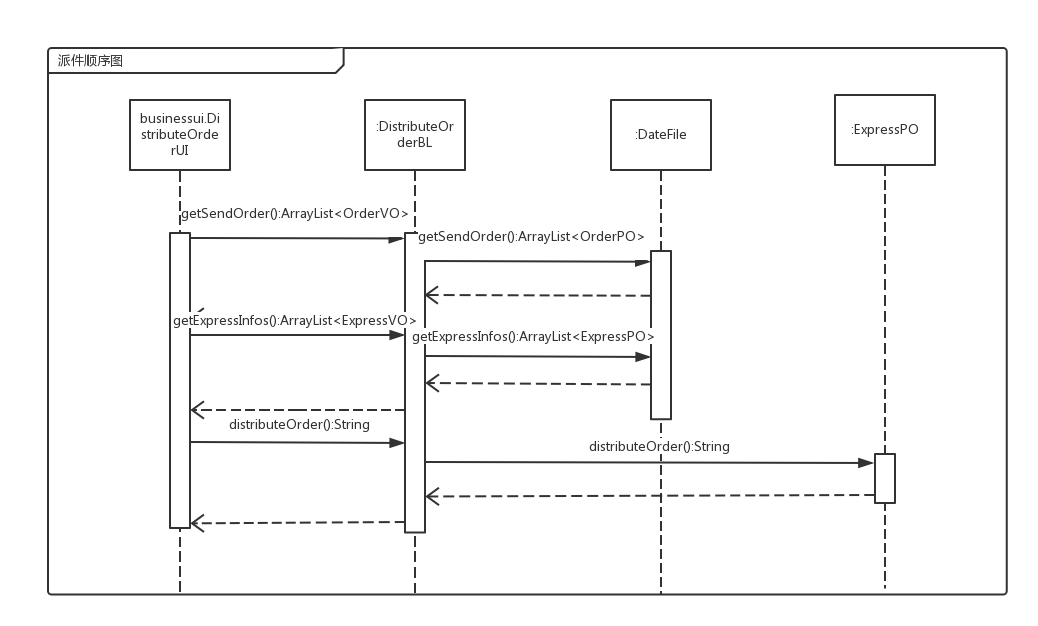
**图2-4 司机信息管理的顺序图**

图2-5为 VehicleManagerBL领域对象想要车辆信息管理时的顺序图



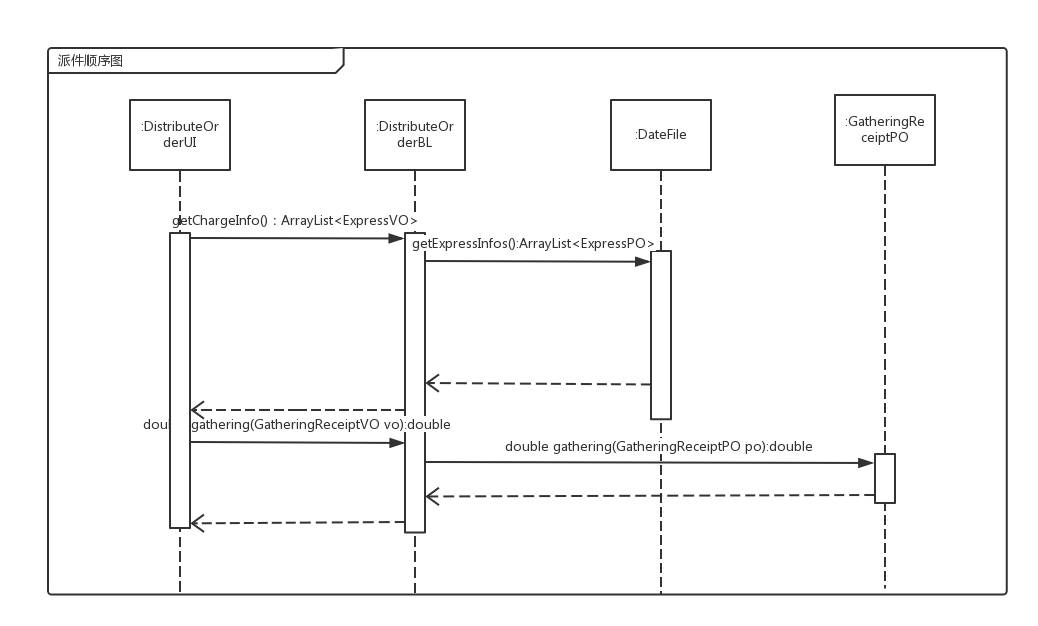
**图2-5 车辆信息管理的顺序图**

图2-6为 DistributeOrderBL领域对象想要派件时的顺序图



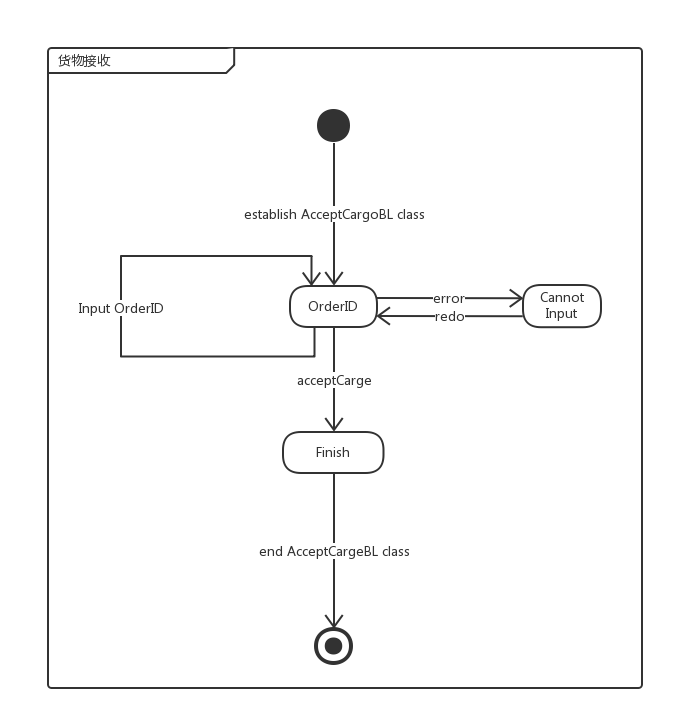
**图2-6 派件的顺序图**

图2-7为 GatheringBL领域对象想要进行收款汇总时的顺序图



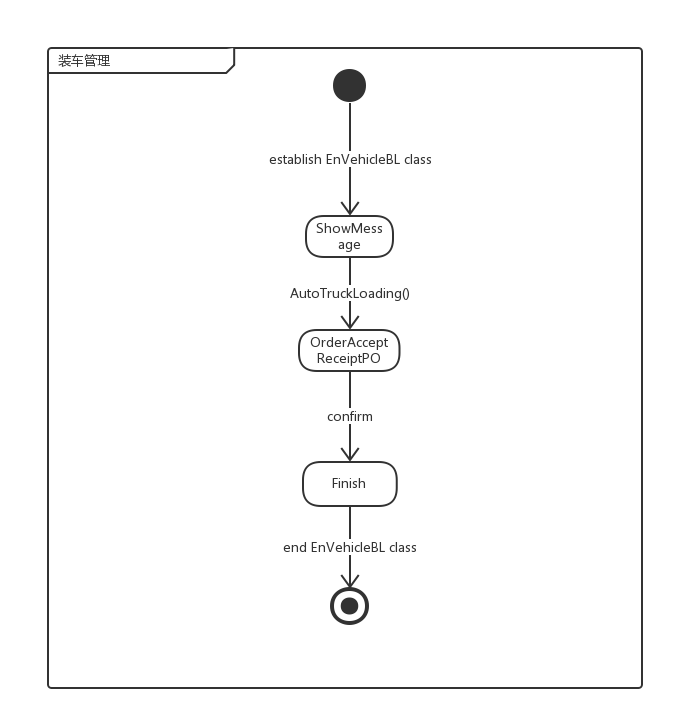
**图2-7 收款汇总的顺序图**

如图1-8所示，AcceptCargoBL对象状态图描述了AcceptCargoBL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establishAcceptCargoBL方法被 UI调用，AcceptCargoBL根据不同的指令进入不同状态。



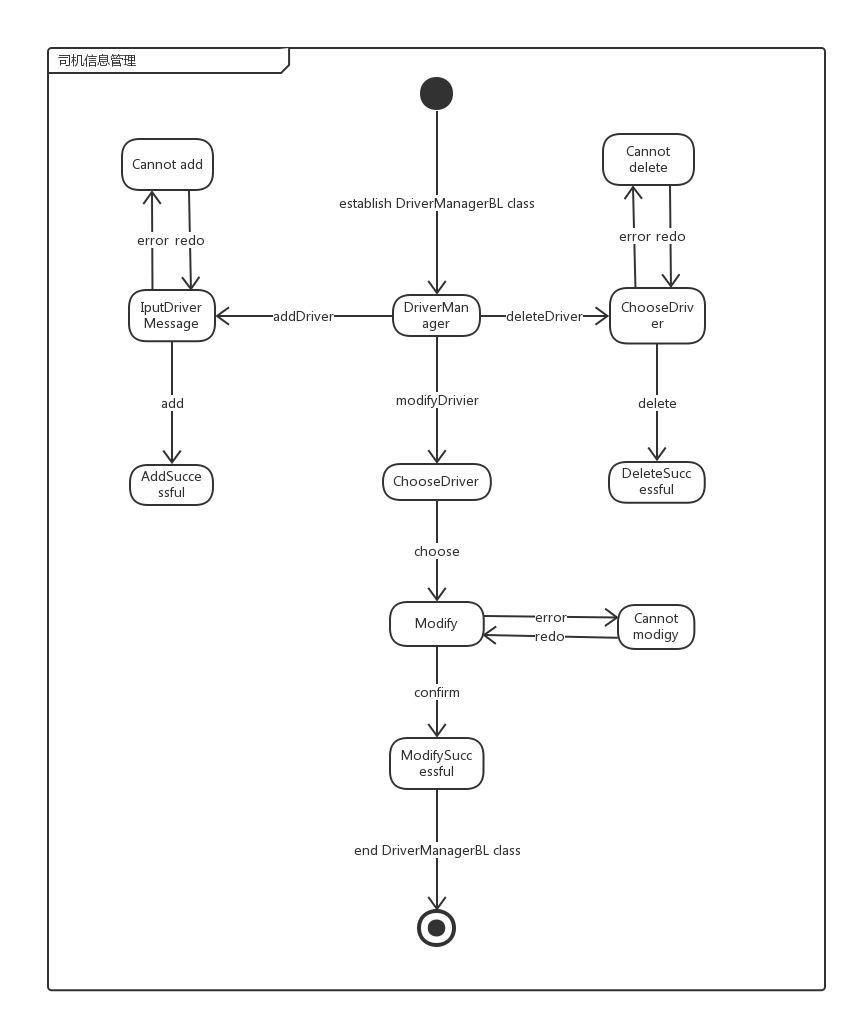
**图1-8 接收货物的状态图**

如图1-9所示，EnVehicleBL对象状态图描述了EnVehicleBL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establishEnVehicleBL方法被 UI调用，EnVehicleBL根据不同的指令进入不同状态。



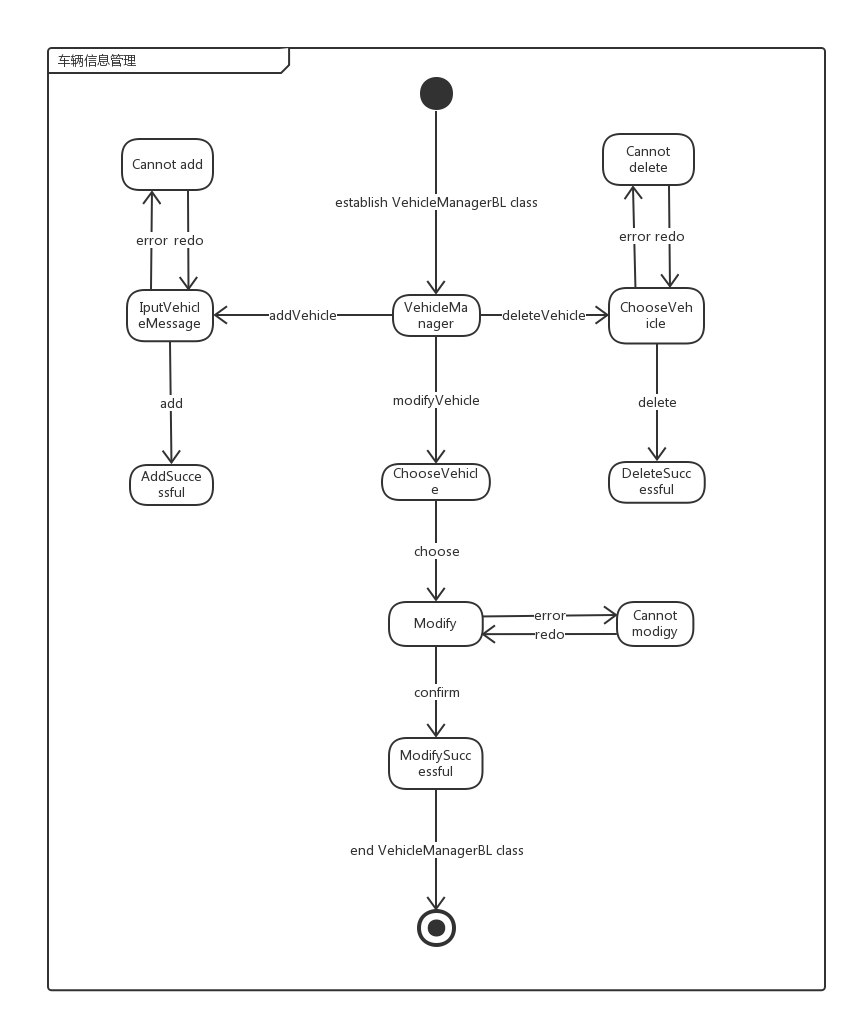
**图1-9 装车管理的状态图**

如图1-10所示，DriverManagerBL对象状态图描述了DriverManagerBL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establishDriverManagerBL方法被 UI调用，DriverManagerBL根据不同的指令进入不同状态。



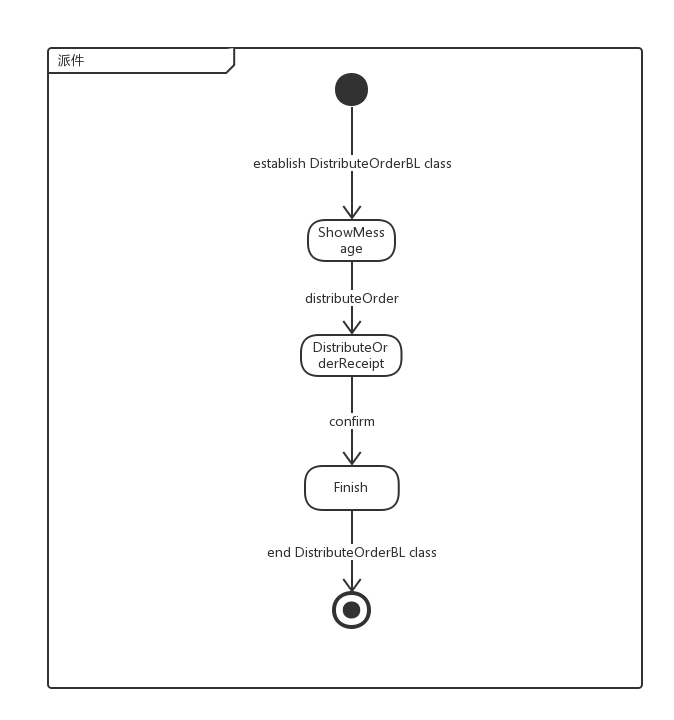
**图1-10 司机信息管理的状态图**

如图1-11所示，VehicleManagerBL对象状态图描述了VehicleManagerBL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establishVehicleManagerBL方法被 UI调用，VehicleManagerBL根据不同的指令进入不同状态。



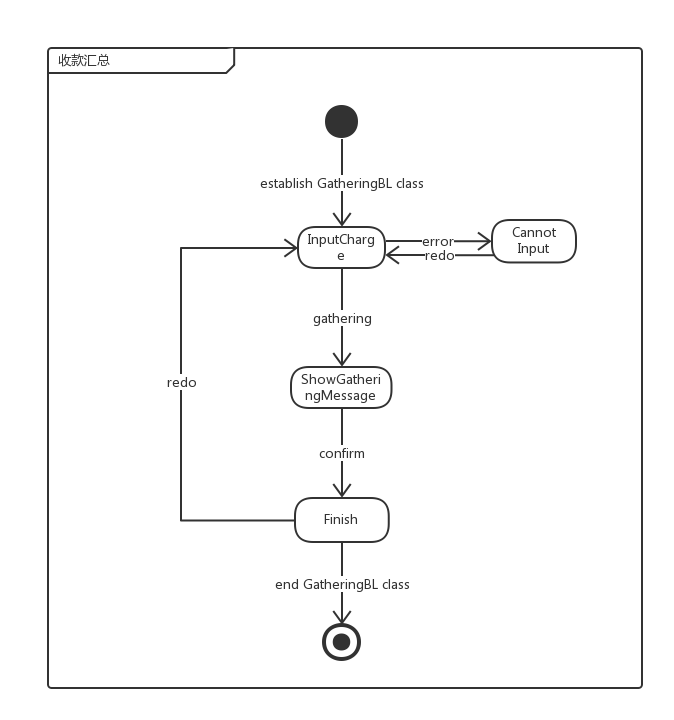
**图1-11 车辆信息管理的状态图**

如图1-12所示，DistributeOrderBL对象状态图描述了DistributeOrderBL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establishDistributeOrderBL方法被 UI调用，DistributeOrderBL根据不同的指令进入不同状态。



**图1-12 派件的状态图**

如图1-13所示，GatheringBL对象状态图描述了GatheringBL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establishGatheringBL方法被 UI调用，GatheringBL根据不同的指令进入不同状态。



**图1-13 收款汇总的状态图**

### 4.2.3intermediatebl模块

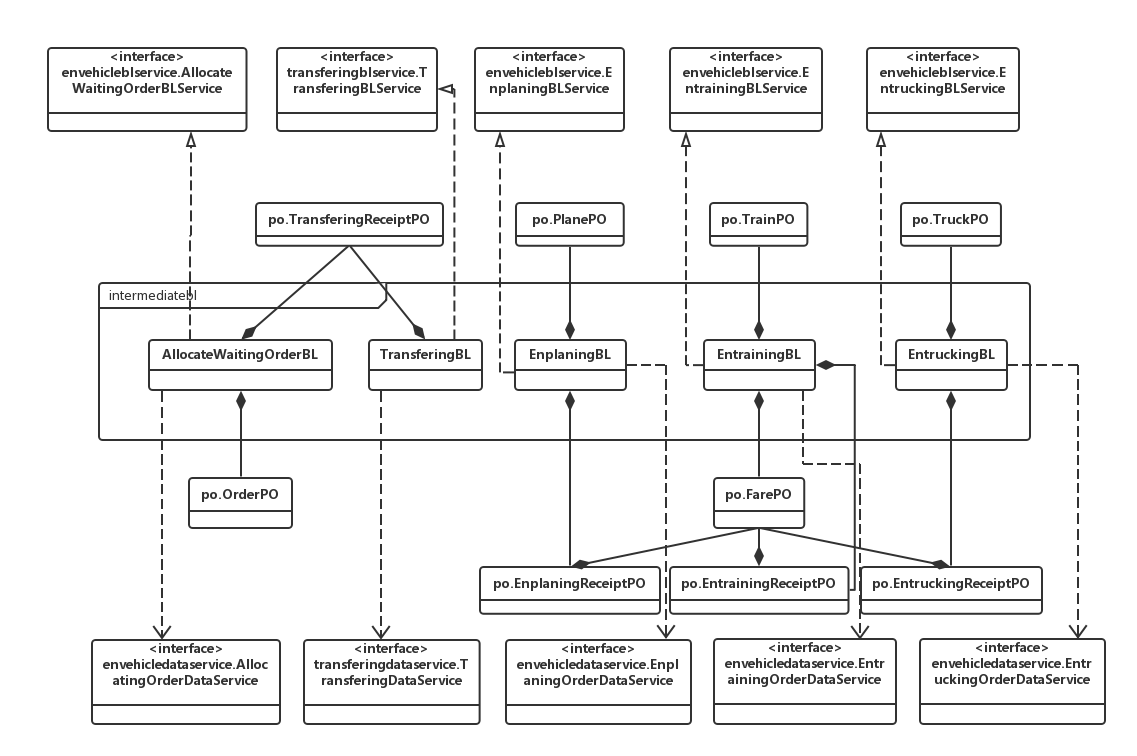
#### 4.2.3.1模块概述

intermediatebl模块承担的需求参见ELS需求规格文档功能需求及相关非功能需求。

intermediatebl模块的职责及接口参见ELS体系结构设计文档。

#### 4.2.3.2整体结构

根据体系结构的设计，我们将系统分为展示层，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。比如展示层和业务逻辑层之间，我们添加businesslogicservice.intermediateblservice.Envehicleblservice,businesslogicservice.intermediateblservice.PlaneManagerBLService,businesslogicservice.intermediateblservice.TrainManagerBLService,businesslogicservice.intermediateblservice.TruckManagerBLService,businesslogicservice.intermediateblservice.AllocateWaitingOrderBLService,businesslogicservice.intermediateblservice.TransferingBLService接口。业务逻辑层和数据层之间添加dataservice.intermediatedataservice.IntermediateDataService.TransferingDataService接口。TransferingReceiptPO是作为中转中心到达单，PlanePO是作为飞机，TrainPO是作为火车，TruckPO是作为汽车，OrderPO是作为订单，FarePO是作为运费，EnplaningReceiptPO，EntrainingReceiptPO，EntruckingReceiptPO是作为装车单，以上PO分别作为持久化对象被添加到 设计模型中去的。Intermediatebl模块的设计如图3-1所示。



**图3-1 intermediatebl模块各个类的设计**

Intermediatebl模块各个类的职责如表3-1所示。

**表3-1 intermediatebl模块个各类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| AllocateWaitingList | 更新待装车的订单列表 |
| PlaneManager | 飞机信息管理，负责增删改中转中心飞机信息 |
| TrainManager | 火车信息管理，负责增删改中转中心火车信息 |
| TruckManager | 汽车信息管理，负责增删改中转中心汽车信息 |
| Transfering | 中转中心负责中转接收的操作并生成中转中心到达单 |
| Envehicle | 中转中心负责装车操作并生成中转中心装车单 |

#### 4.2.3.3模块内部类的接口规范

AllocateWaitingList，PlaneManager，TrainManager，TruckManager，Envehicle和Transfering的接口规范如表3-2，3-3，3-4，3-5和3-6所示。

**表3-2 AllocateWaitingList的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Intermediate.updateWaitingList | 语法 | public ArrayList<OrderVO> updateWaitingList() |
| 前置条件 | 快件被装车或中转中心到达单更新 |
| 后置条件 | 等候装车的快件列表更新 |
| Intermediate.showOrder | 语法 | public OrderVO showOrder(String orderID) throws Exception |
| 前置条件 | 等候装车订单列表中某订单被选择 |
| 后置条件 | 显示订单信息 |
| 需要的服务（需接口） | | |  | 快件被装车或中转中心到达单更新 |
| 服务名 | | 服务 |  |
| IntermediateDataService.updateWaitingList(TranferingReceiptPO) | | 更新一个存储OrderPO对象的List对象 |

**表3-3 PlaneManager的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Intermediate.showPlaneList | 语法 | public ArrayList<PlaneVO> showPlaneList(); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心飞机列表的权利，并选择查看所有飞机信息列表 |
| 后置条件 | 系统根据中转中心编号正确显示该中转中心飞机信息列表 |
| Intermediate. showPlane | 语法 | public PlaneVO showPlane(String ID); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心某飞机当前包括已装车快件信息列表在内的所有飞机属性信息的权利，并正确输入了飞机编号或正确选择了飞机 |
| 后置条件 | 系统根据输入的飞机编号显示该飞机信息 |
| Intermediate. addPlane | 语法 | public OperationState addPlane(String ID, String destination) throws RemoteException |
| 前置条件 | 经过验证的中转中心业务员有新增飞机的权利，选择新增飞机功能 |
| 后置条件 | 新增飞机进入本中转中心飞机列表中 |
| Intermediate. DeletePlane | 语法 | public OperationState deletePlane(String s) throws Exception; |
| 前置条件 | 经过验证的中转中心业务员有删除飞机的权利，选择已有飞机从飞机列表中删除 |
| 后置条件 | 删除飞机不在本中转中心飞机列表中 |
| Intermediate.modifyPlane | 语法 | public OperationState modifyPlane(PlaneVO plane\_modify) throws Exception; |
| 前置条件 | 经过验证的中转中心业务员有修改飞机的权利，选择已有飞机在飞机列表中修改 |
| 后置条件 | 飞机信息修改完毕并更新至飞机列表中 |
| Intermediate. savePlaneList | 语法 | public OperationState savePlaneList() throws RemoteException; |
| 前置条件 | 中转中心业务员确认飞机列表正确并保存 |
| 后置条件 | 飞机列表文件已保存在中转中心文件中 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| IntermediateDataService.getPlaneList() | | 返回一个存储PlanePO的List对象 |
| IntermediateDataService. getPlane(String ID) | | 返回一个PlanePO对象 |
| IntermediateDataService.savePlaneList(ArrayList<PlanePO> list) | | 存储一个包含PlanePO的List对象 |
| IntermediateDataService. savePlane(PlanePO po) | | 在用户数据文件中插入一个PlanePO对象 |

**表3-4 TrainManager的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Intermediate.showTrainList | 语法 | public ArrayList<TrainVO> showTrainList(); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心火车列表的权利，并选择查看所有火车信息列表 |
| 后置条件 | 系统根据中转中心编号正确显示该中转中心火车信息列表 |
| Intermediate. showTrain | 语法 | public TrainVO showTrain(String ID); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心某火车当前包括已装车快件信息列表在内的所有火车属性信息的权利，并正确输入了火车编号或正确选择了火车 |
| 后置条件 | 系统根据输入的火车编号显示该火车信息 |
| Intermediate. addTrain | 语法 | public OperationState addTrain(String ID, String destination) throws RemoteException |
| 前置条件 | 经过验证的中转中心业务员有新增火车的权利，选择新增火车功能 |
| 后置条件 | 新增火车进入本中转中心火车列表中 |
| Intermediate. DeleteTrain | 语法 | public OperationState deleteTrain(String s) throws Exception; |
| 前置条件 | 经过验证的中转中心业务员有删除火车的权利，选择已有火车从火车列表中删除 |
| 后置条件 | 删除火车不在本中转中心火车列表中 |
| Intermediate.modifyTrain | 语法 | public OperationState modifyTrain(TrainVO Train\_modify) throws Exception; |
| 前置条件 | 经过验证的中转中心业务员有修改火车的权利，选择已有火车在火车列表中修改 |
| 后置条件 | 火车信息修改完毕并更新至火车列表中 |
| Intermediate. saveTrainList | 语法 | public OperationState saveTrainList() throws RemoteException; |
| 前置条件 | 中转中心业务员确认火车列表正确并保存 |
| 后置条件 | 火车列表文件已保存在中转中心文件中 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| IntermediateDataService.getTrainList() | | 返回一个存储TrainPO的List对象 |
| IntermediateDataService. getTrain(String ID) | | 返回一个TrainPO对象 |
| IntermediateDataService.saveTrainList(ArrayList<TrainPO> list) | | 存储一个包含TrainPO的List对象 |
| IntermediateDataService. saveTrain(TrainPO po) | | 在用户数据文件中插入一个TrainPO对象 |

**表3-5 TruckManager的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Intermediate.showTruckList | 语法 | public ArrayList<TruckVO> showTruckList(); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心汽车列表的权利，并选择查看所有汽车信息列表 |
| 后置条件 | 系统根据中转中心编号正确显示该中转中心汽车信息列表 |
| Intermediate. showTruck | 语法 | public TruckVO showTruck(String ID); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心某汽车当前包括已装车快件信息列表在内的所有汽车属性信息的权利，并正确输入了汽车编号或正确选择了汽车 |
| 后置条件 | 系统根据输入的汽车编号显示该汽车信息 |
| Intermediate. addTruck | 语法 | public OperationState addTruck(String ID, String destination) throws RemoteException |
| 前置条件 | 经过验证的中转中心业务员有新增汽车的权利，选择新增汽车功能 |
| 后置条件 | 新增汽车进入本中转中心汽车列表中 |
| Intermediate. DeleteTruck | 语法 | public OperationState deleteTruck(String s) throws Exception; |
| 前置条件 | 经过验证的中转中心业务员有删除汽车的权利，选择已有汽车从汽车列表中删除 |
| 后置条件 | 删除汽车不在本中转中心汽车列表中 |
| Intermediate.modifyTruck | 语法 | public OperationState modifyTruck(TruckVO Truck\_modify) throws Exception; |
| 前置条件 | 经过验证的中转中心业务员有修改汽车的权利，选择已有汽车在汽车列表中修改 |
| 后置条件 | 汽车信息修改完毕并更新至汽车列表中 |
| Intermediate. saveTruckList | 语法 | public OperationState saveTruckList() throws RemoteException; |
| 前置条件 | 中转中心业务员确认汽车列表正确并保存 |
| 后置条件 | 汽车列表文件已保存在中转中心文件中 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| IntermediateDataService.getTruckList() | | 返回一个存储TruckPO的List对象 |
| IntermediateDataService. getTruck(String ID) | | 返回一个TruckPO对象 |
| IntermediateDataService.saveTruckList(ArrayList<TruckPO> list) | | 存储一个包含TruckPO的List对象 |
| IntermediateDataService. saveTruck(TruckPO po) | | 在用户数据文件中插入一个TruckPO对象 |

**表3-6 Transfering的接口规范**

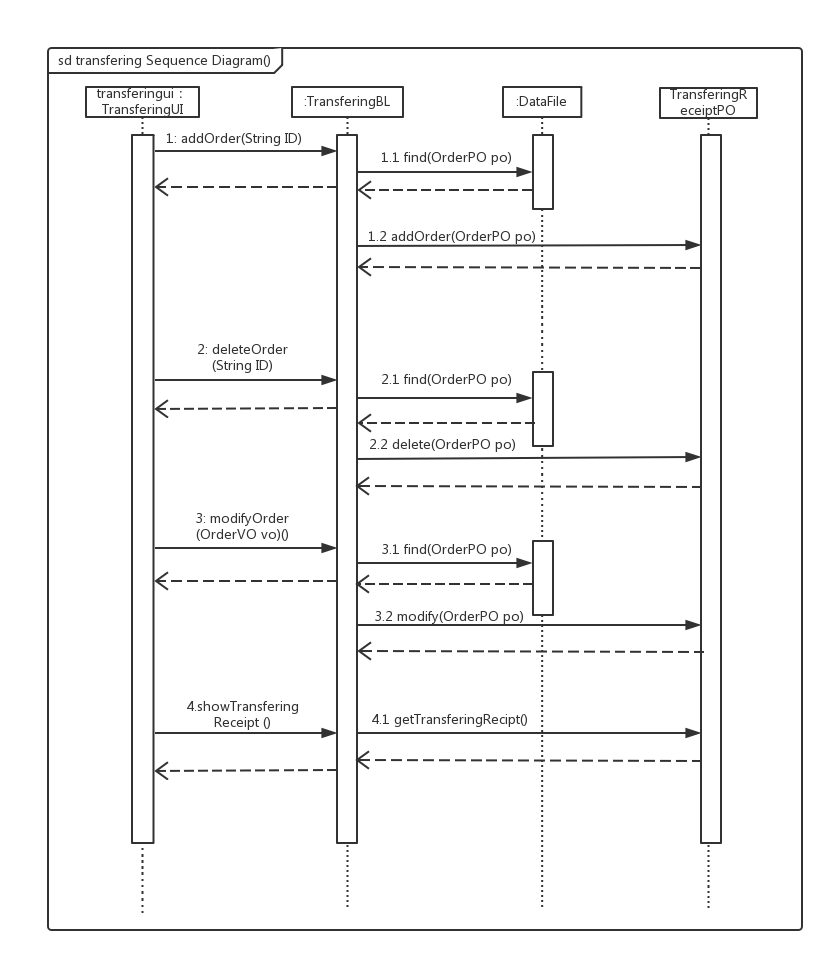
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Intermediate. showTransferingReceipt | 语法 | | Public TransferingReceiptVO showTransferingReceipt(); |
| 前置条件 | | 经过验证的中转中心业务员有查看中转中心到达单的权利，且业务员选择了中转中心到达单查看功能 |
| 后置条件 | | 系统根据日期和中转中心编号显示正确的中转中心到达单 |
| Intermediate.addOrder | 语法 | | public boolean addOrder(String ID); |
| 前置条件 | | 经过验证的中转中心业务员有添加新寄件单到中转中心到达单的权利，且业务员输入了完整有效的订单号 |
| 后置条件 | | 系统保存新增的寄件单信息并更新至中转中心到达单 |
| Intermediate.deleteOrder | 语法 | | public boolean deleteOrder(String ID); |
| 前置条件 | | 经过验证的中转中心业务员有删除新寄件单从中转中心到达单的权利，且业务员输入了完整有效的订单号或选择了正确的删除对象 |
| 后置条件 | | 系统查找对应的订单信息，并删除该用户对象 |
| Intermediate.saveTransferingReceipt | 语法 | | public void updateTransferingReceipt(TransferingReceiptVO); |
| 前置条件 | | 经过验证的中转中心业务员选择更新中转中心到达单 |
| 后置条件 | | 系统根据中转中心到达单将中转中心到达单数据存储至数据库中 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FileFactory.getIntermediateData | | 得到Intermediate文件的引用 | |
| IntermediateDataService. showTransferingReceipt（） | | 返回一个TransferingReceiptPO对象 | |
| IntermediateDataService.addOrder(String ID) | | 在用户数据文件中插入一个OrderPO对象 | |
| IntermediateDataService.deleteOrder(String ID) | | 在用户数据文件中删除一个OrderPO对象 | |
| IntermediateDataService.modifyOrder(String ID) | | 在用户数据文件中更新一个OrderPO对象 | |
| IntermediateDataService.updateTransferingReceipt(TransferingReceiptPO) | | 在用户数据文件中插入一个TransferingReceiptPO对象 | |
| IntermediateDataService. saveTruck(TruckPO po) | | 在用户数据文件中插入一个TruckPO对象 | |

**表3-7 Envehicle的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Intermediate. Envehicle | 语法 | | public OperationState envehicle() throws Exception |
| 前置条件 | | 经过验证的中转中心业务员有装车管理的权利，且业务员选择了装车管理功能 |
| 后置条件 | | 系统自动进行一键装车操作 |
| Intermediate.saveEnIntermediateReceiptList | 语法 | | public OperationState saveEnIntermediateReceiptList()  throws RemoteException |
| 前置条件 | | 经过验证的中转中心业务员有存储装车单的权利，且业务员选择了保存装车单的功能 |
| 后置条件 | | 系统保存新增的装车单信息并更新至数据文件中 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| IntermediateDataService. saveEnIntermediateReceiptList | | 在数据文件中保存中转中心装车单信息 | |

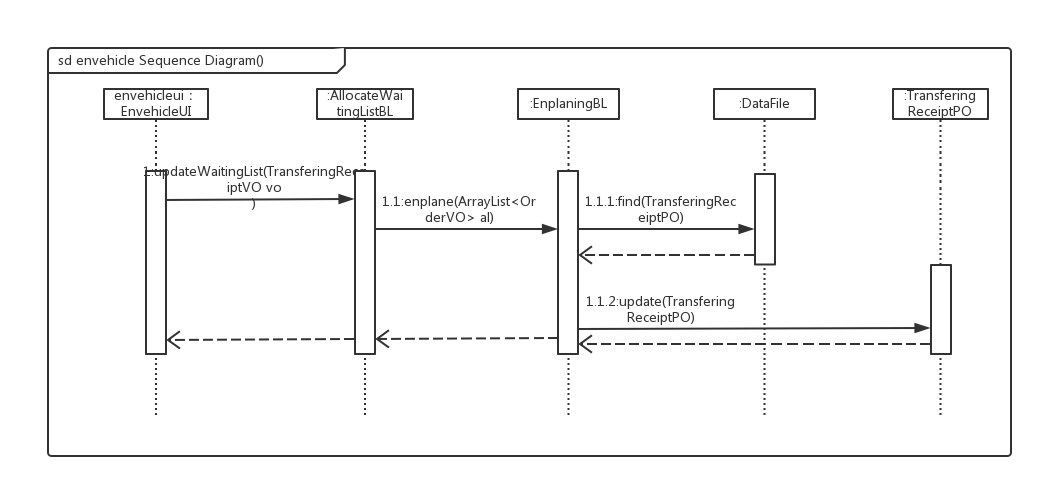
#### 4.2.3.4业务逻辑层的动态模型

图3-2表明了ELS快递物流系统中，当中转中心业务员选择中转接收功能并输入新增中转中心到达单上的底单之后，中转接收业务逻辑处理的相关对象之间的协作。



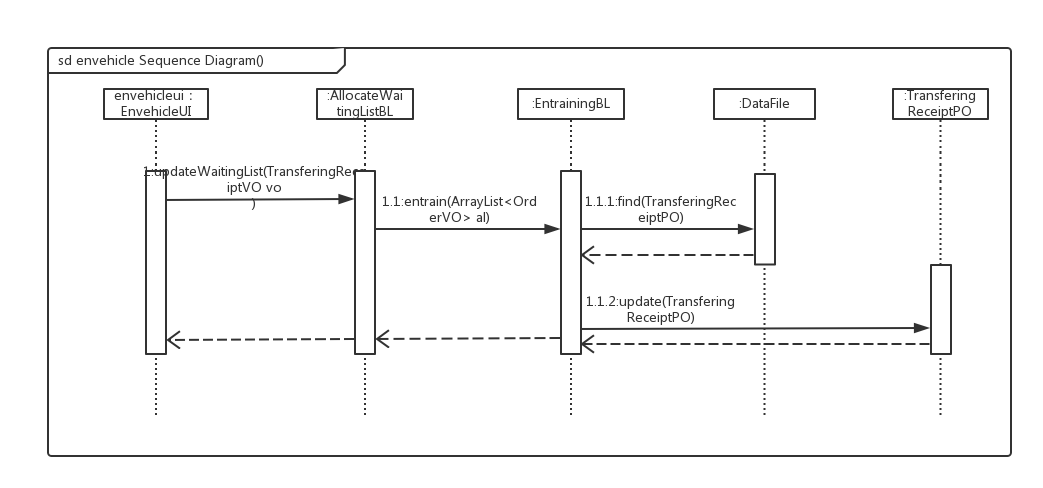
**图3-2 中转接收的顺序图**

图3-3，3-4，3-5表明了ELS快递物流系统中，当用户选择飞机，火车，汽车装运管理并更新待装车订单列表之后，飞机，火车，汽车装运管理业务逻辑处理的相关对象之间的协作。

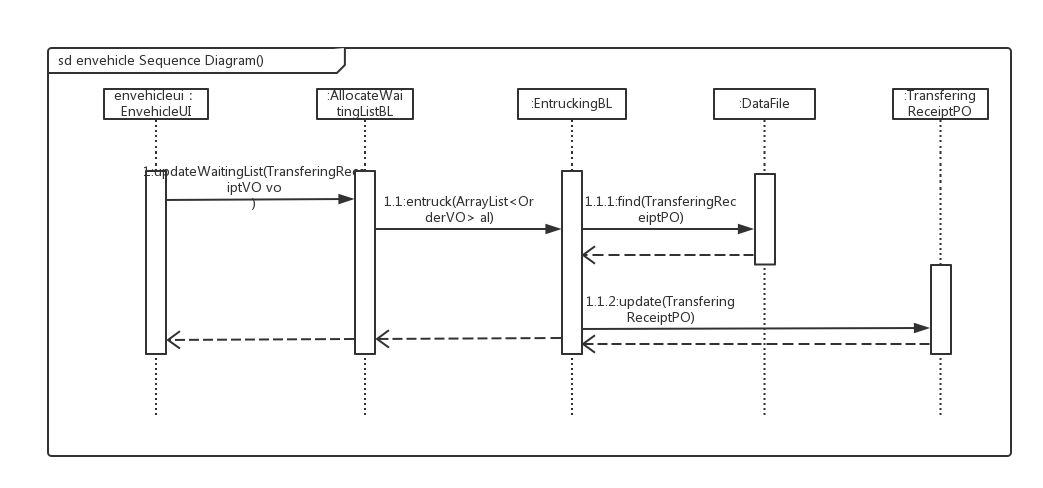


**图3-3 飞机装运管理的顺序图**

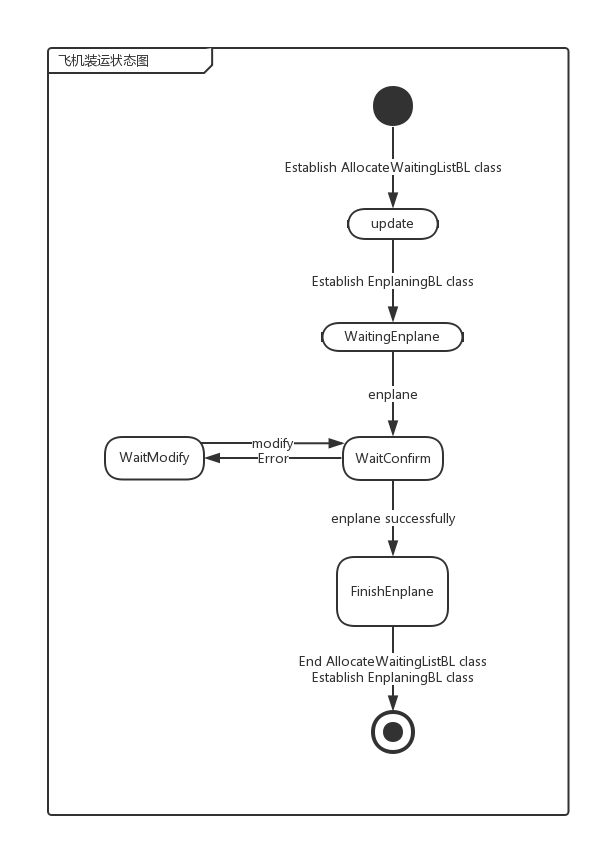
**图3-4 火车装运管理的顺序图**



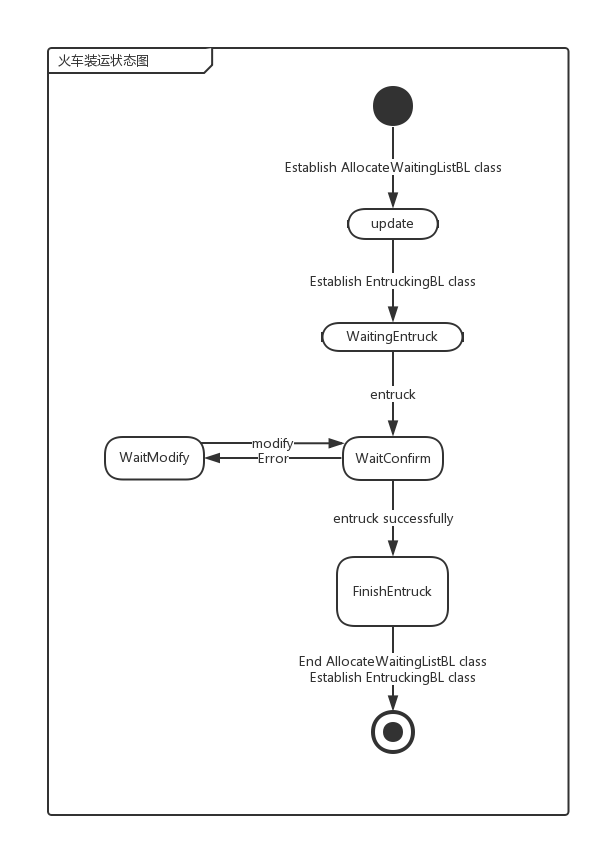
**图3-5 汽车装运管理的顺序图**



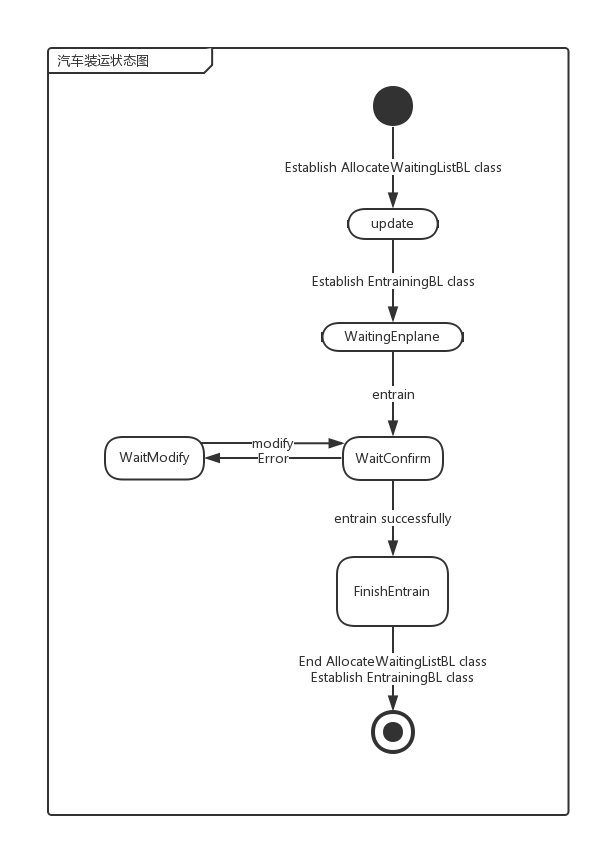
如图3-6，3-7，3-8所示，装运管理对象状态图分别描述了Enplaning，Entraining，Entrucking对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着establishAllocateWaitingList方法以及establishEnplaning，establishEntraining，establishEntrucking被 UI调用，Enplaning，Entraining，Entrucking根据不同的指令进入不同状态。



**图3-6 飞机装运管理的状态图**

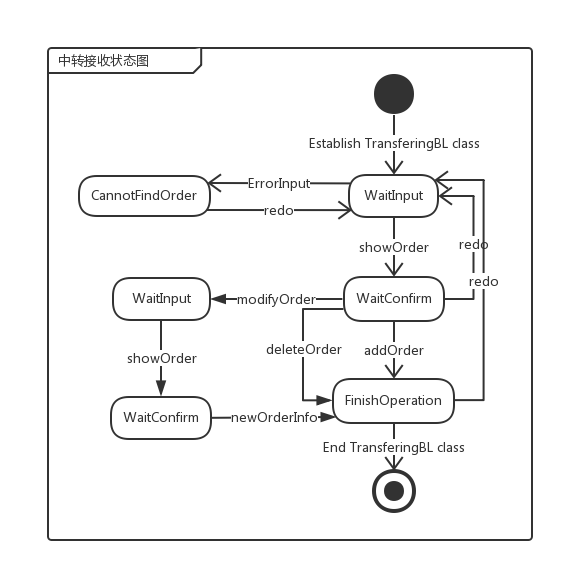


**图3-7 火车装运管理的状态图**



**图3-8 汽车装运管理的状态图**

如图3-9所示，中转接收对象状态图描述了Transfering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establishTransfering方法被 UI调用，establishTransfering根据不同的指令进入不同状态。



**图3-9 中转接收的状态图**

### 4.2.4receiptbl模块

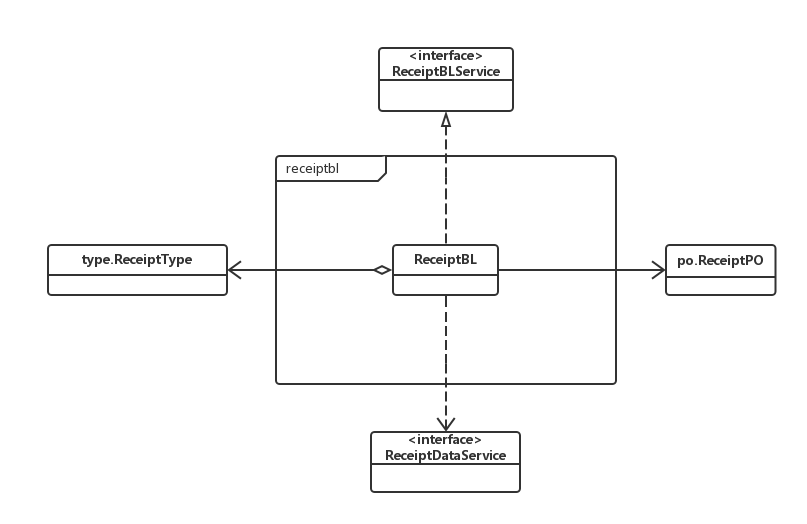
#### 4.2.4.1模块概述

receiptbl模块承担的需求参见ELS需求规格文档功能需求及相关非功能需求。

receiptbl模块的职责及接口参见ELS体系结构设计文档。

#### 4.2.4.2整体结构

根据体系结构的设计，我们将系统分为展示层，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。比如展示层和业务逻辑层之间，我们添加receiptblservice.ReceiptBLService接口。业务逻辑层和数据层之间添加receiptdataservice.ReceiptDataService接口。ReceiptPO是作为所有单据父类的持久化对象被添加到设计模型中去的。ReceiptType是作为存放所有ReceiptPO当前状态的枚举被添加到设计模型中去的。ReceiptBL模块的设计如图4-1所示。



**图4-1 receiptbl模块各个类的设计**

receiptbl模块各个类的职责如表4-1所示。

**表4-1 receiptbl模块个各类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ReceiptBL | 负责所有单据基本增删改查单据属性的操作 |

#### 4.2.4.3模块内部类的接口规范

ReceiptBL的接口规范如表4-2所示。

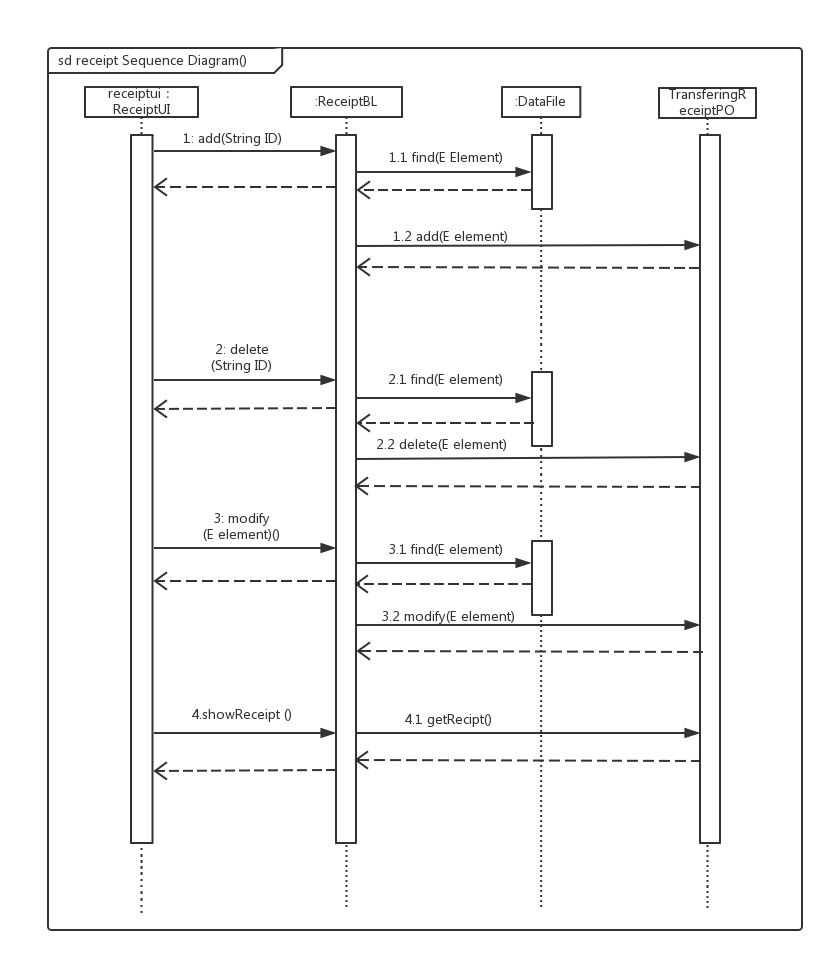
**表4-2 ReceiptBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| Receipt.add | 语法 | Public int add(ReceiptVo vo) | |
| 前置条件 | User经验证，可以创建单据 | |
| 后置条件 | 向待审批单据列表库中添加该单据 | |
| Receipt.modify | 语法 | Public int modify(String ID) | |
| 前置条件 | User选中了有效单据 | |
| 后置条件 | 更新该单据信息 | |
| Receipt.batch | 语法 | Public int batch(String[] ID) | |
| 前置条件 | 当前有待审批单据，且user已选中单据 | |
| 后置条件 | 更新该单据组信息 | |
| Receipt.update | 语法 | Public void update(ReceiptVo vo) | |
| 前置条件 | 该业务单据创建成功 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Receipt.reply | 语法 | Public void reply(String userID) | |
| 前置条件 | 该单据审批完成 | |
| 后置条件 | 该单据制单人收到审批结果信息 | |
| Receipt.view | 语法 | Public ArrayList<ReceiptVO> view() | |
| 前置条件 | 已有单据记录 | |
| 后置条件 | 返回所有单据信息 | |
| Receipt.refresh | 语法 | Public ArrayList<ReceiptVO> refresh() | |
| 前置条件 | 已有单据记录 | |
| 后置条件 | 返回所有待审批单据信息 | |
| Receipt.approve | 语法 | Public int approve(String ID) | |
| 前置条件 | 有单据提交待审批 | |
| 后置条件 | 更新该单据信息 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 |
| FileFactory.getReceiptData | | | 得到Receipt文件的引用 |
| ReceiptDataService.add(ReceiptPo po) | | | 插入单一持久化对象 |
| ReceiptDataService.modify(ReceiptPo po) | | | 更新单一持久化对象 |
| ReceiptDataService.modify(ReceiptPo[] po) | | | 更新单一持久化对象组 |
| ReceiptDataService.find(Type type) | | | 根据类型type查找ReceiptPo对象 |
| ReceiptDataService.find(Date date) | | | 根据日期date查找ReceiptPo对象 |
| ReceiptDataService.show() | | | 得到所得的ReceiptPo对象 |

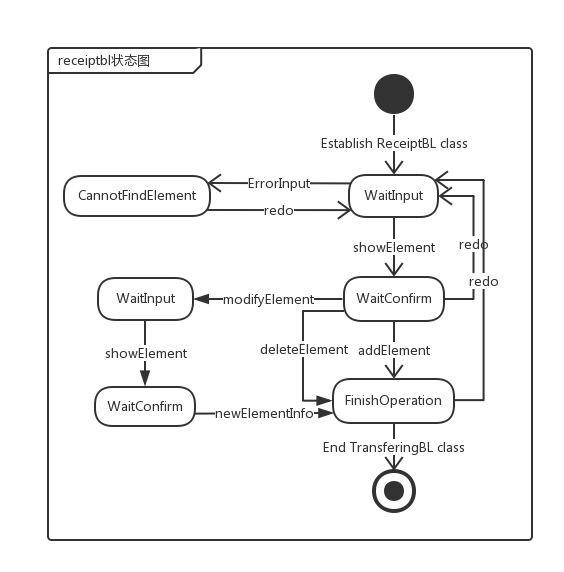
#### 4.2.4.4业务逻辑层的动态模型

图4-2表明了ELS快递物流系统中，当选择单据增删改查的基本操作之后，单据业务逻辑处理的相关对象之间的协作。

**图4-2 单据增删改查的顺序图**



如图4-3所示，单据增删改查对象状态图描述了Receipt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establishReceipt方法被 UI调用，establishReceipt根据不同的指令进入不同状态。



**图4-3 单据增删改查的状态图**

### 4.2.5repertorybl模块

#### 4.2.5.1模块概述

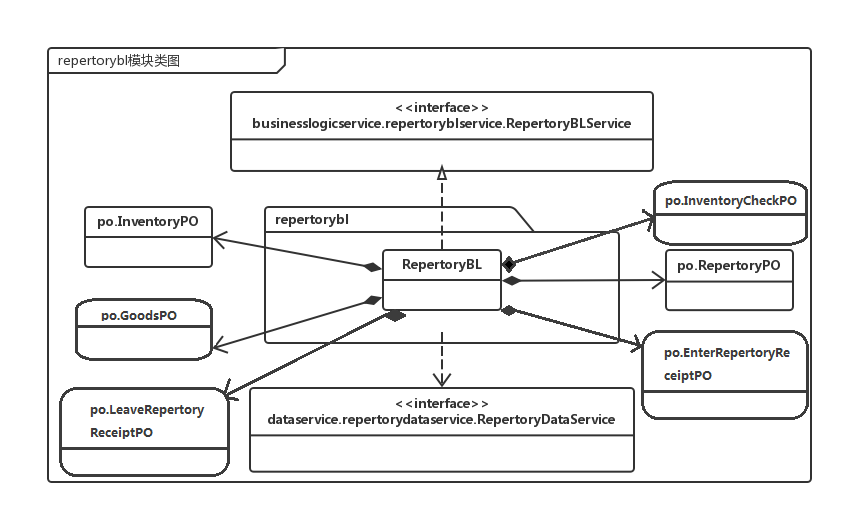
repertorybl模块承担的需求参见ELS需求规格说明文档功能需求及相关非功能需求。

repertorybl模块的职责及接口参见ELS体系结构设计文档表13。

#### 4.2.5.2整体结构

根据体系结构的设计，我们将系统分为展示层、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。比如展示层和业务逻辑层之间，我们添加businesslogicservice.repertoryblservice.RepertoryBLService接口。业务逻辑层和数据层之间添加dataservice.repertorydataservice.RepertoryDataService接口。因为业务逻辑处理较少，根据专家模式没有添加Controller类。RepertoryPO等PO对象是作为仓库管理员管理记录的持久化对象被添加到设计模型中去的。

repertorybl模块的设计如图5-1所示。



**图5-1 repertorybl模块各个类的设计**

repertorybl模块各个类的职责如表5-1所示。

**表5-1 repertorybl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| RepertoryBL | 仓库的领域模型对象，拥有仓库数据的编号、所属管理员编号、最大容量、警戒比例，可以解决仓库管理员仓库管理的问题。 |
| GoodsBL | 仓库的领域模型对象，拥有仓库中库存的编号、位置、货物的历史轨迹、可以解决仓库管理员仓库管理的问题。 |
| EnterRepertoryReceiptBL | 入库单的领域模型对象，拥有入库单的编号、创建时间、制单人、入库单类别、入库单状态、入库货物信息列表，可以解决仓库管理员仓库管理的问题。 |
| LeaveRepertoryReceiptBL | 出库单的领域模型对象，拥有出库单的编号、创建时间、制单人、出库单类别、出库单状态、出库货物信息列表，可以解决仓库管理员仓库管理的问题。 |

**4.2.5.3模块内部类的接口规范**

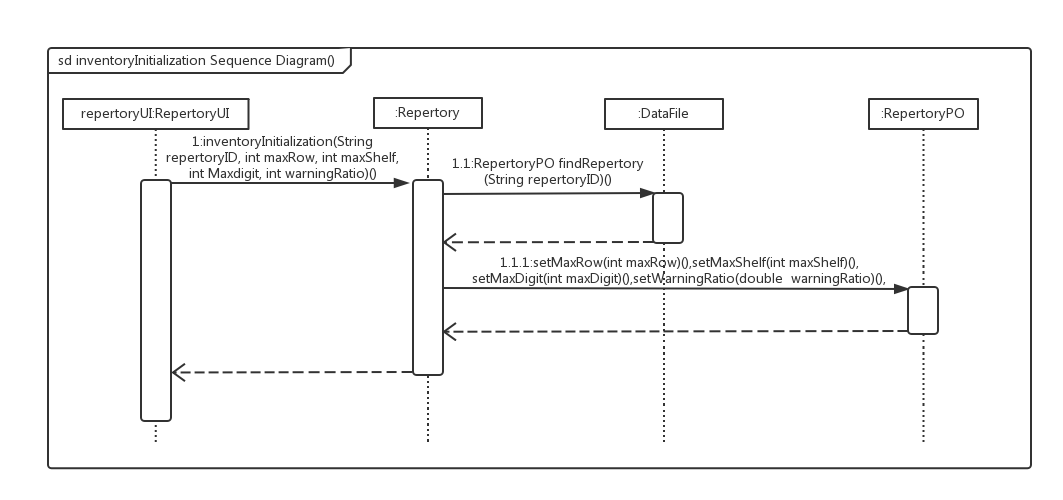
RepertoryBL的接口规范如表5-2所示。

**表5-2 RepertoryBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Repertory.inventoryInitialization | 语法 | | public int inventoryInitialization(int maxRow, int maxShelf, int Maxdigit, int warningRatio) |
| 前置条件 | | 经过验证的仓库管理员有库存信息初始化的权利，且仓库管理员输入了有效的库存信息值 |
| 后置条件 | | 系统保存仓库初始化信息 |
| Repertory.enterRepertory | 语法 | | public int enterRepertory(String orderID, int blockNum, int rowNum, int shelfNum, int digitNum) |
| 前置条件 | | 经过验证的仓库管理员有将商品入库的权利，且仓库管理员输入了有效的商品信息和对应仓库位置信息、系统自动获取时间信息并登记 |
| 后置条件 | | 系统保存商品入库的位置和时间信息，生成入库单 |
| Repertory.leaveRepertory | 语法 | | public int leaveRepertory(String orderID) |
| 前置条件 | | 经过验证的仓库管理员有将商品出库的权利，且仓库管理员输入了有效的商品信息和转运方式、时间信息 |
| 后置条件 | | 系统保存商品出库的时间，生成出库单 |
| Repertory.inventoryWarning | 语法 | | public boolean inventoryWarning() |
| 前置条件 | | 当某区的库存总量与某区最大容量的比值达到或者超过警戒比例时，系统提示仓库管理员该区库存已达警戒比例 |
| 后置条件 | | 无 |
| Repertory.inventoryCheck | 语法 | | public InventoryCheckVO inventoryCheck(String beginDate, String endDate) |
| 前置条件 | | 经过验证的仓库管理员有库存查看的权利，且仓库管理员输入了有效的时间段信息 |
| 后置条件 | | 系统根据输入的时间段查找对应的出入库商品，并返回这段时间出/入库商品数量和金额总和，库存数量合计 |
| Repertory.inventoryStockTaking | 语法 | | public ArrayList<InventoryVO> inventoryStockTaking() |
| 前置条件 | | 经过验证的仓库管理员有库存盘点的权利，且仓库管理员输入了库存盘点命令 |
| 后置条件 | | 系统自动根据提交库存盘点命令的时间生成一个截止点，并根据该截止点查找当天到截止点之前的仍在仓库中的商品，返回商品信息 |
| Repertory.getLastEnterRepertoryTime | 语法 | | public String getLastEnterRepertoryTime() |
| 前置条件 | | 经过验证的仓库管理员有生成入库单的权利，系统显示上一次生成入库单的时间，用来提示仓库管理员 |
| 后置条件 | | 无 |
| Repertory.getLastLeaveRepertoryTime | 语法 | | public String getLastLeaveRepertoryTime() |
| 前置条件 | | 经过验证的仓库管理员有生成出库单的权利，系统显示上一次生成出库单的时间，用来提示仓库管理员 |
| 后置条件 | | 无 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FileFactory.getRepertoryData | | 得到Repertory文件的引用 | |
| FileFactory.getGoodsData | | 得到Goods文件的引用 | |
| FileFactory.getEnterRepertoryReceiptData | | 得到EnterRepertoryReceipt文件的引用 | |
| FileFactory.getLeaveRepertoryReceiptData | | 得到LeaveRepertoryReceipt文件的引用 | |
| GoodsDataService.addGoodss(GoodssPO Goodsspo) | | 插入单一持久化对象GoodsPO对象 | |
| GoodsDataService.deleteGoodss(String orderID) | | 删除单一持久化对象GoodsPO对象 | |
| GoodsDataService.modifyGoodss(GoodssPO Goodsspo) | | 修改单一持久化对象GoodsPO对象 | |
| GoodsDataService.findGoodssbyID(String orderID) | | 根据输入的订单号，返回对应的GoodsPO对象 | |
| GoodsDataService.findGoodssbyKeyword(String keyword) | | 根据输入的关键字，返回包含关键字的对应的GoodsPO对象列表 | |
| GoodsDataService.getAllFreeGoodss() | | 得到所有的不在仓库内的货物列表 | |
| GoodsDataService.findFreeGoodssByKeyword(String keyword) | | 根据输入的关键字，返回所有的包含关键字的不在仓库内的货物列表 | |
| RepertoryDataService.modifyRepertory(RepertoryPO repertorypo) | | 修改单一持久化对象RepertoryPO对象 | |
| RepertoryDataService.findRepertory(String repertoryID) | | 根据输入的仓库编号，返回对应的RepertoryPO对象 | |
| RepertoryDataService.findRepertoryByOwnerID(String ownerID) | | 根据输入的仓库管理员编号，返回该仓库管理员对应负责的RepertoryPO对象 | |
| RepertoryDataService. showAllRepertorys() | | 得到所有的仓库列表 | |
| RepertoryDataService. addInventory(String repertoryID, InventoryPO inventorypo) | | 在编号为repertoryID的仓库的库存列表中插入单一持久化对象InventoryPO对象 | |
| RepertoryDataService. deleteInventory(String repertoryID, String orderID) | | 在编号为repertoryID的仓库的库存列表中删除单一持久化对象InventoryPO对象 | |
| RepertoryDataService. modifyInventory(String repertoryID, InventoryPO inventorypo) | | 在编号为repertoryID的仓库的库存列表中修改单一持久化对象InventoryPO对象 | |
| RepertoryDataService. findInventorybyID(String repertoryID, String orderID) | | 根据输入的仓库编号和货物编号，返回对应的InventoryPO对象 | |
| RepertoryDataService. findInventorybyDate(String repertoryID, String beginDate, String endDate) | | 根据输入的仓库编号和起始时间，返回对应的InventoryCheckPO对象 | |
| RepertoryDataService. findInventorybyTime(String repertoryID, String time) | | 根据输入的仓库编号和截止时间，返回对应的InventoryPO对象列表 | |
| RepertoryDataService. getLastCreateEnterReceiptTime(String repertoryID) | | 根据输入的仓库编号，返回编号为repertoryID的仓库上一次生成入库单的时间 | |
| RepertoryDataService. getLastCreateLeaveReceiptTime(String repertoryID) | | 根据输入的仓库编号，返回编号为repertoryID的仓库上一次生成出库单的时间 | |
| RepertoryDataService. getEnterRepertoryGoods(String repertoryID) | | 根据输入的仓库编号，返回编号为repertoryID的仓库在上一次生成出库单后到现在这段时间内进入过该仓库的所有货物列表 | |
| RepertoryDataService. getLeaveRepertoryGoods(String repertoryID) | | 根据输入的仓库编号，返回编号为repertoryID的仓库在上一次生成出库单后到现在这段时间内离开过该仓库的所有货物列表 | |
| EnterRepertoryReceiptDataService.addEnterRepertoryReceipt(EnterRepertoryReceiptPO errpo) | | 插入单一持久化对象EnterRepertoryReceiptPO对象 | |
| EnterRepertoryReceiptDataService.sendEnterReceipt(String receiptID) | | 将编号为receiptID的入库单提交至总经理审批 | |
| EnterRepertoryReceiptDataService.approveEnterReceipt(String receiptID) | | 审批通过编号为receiptID的入库单 | |
| EnterRepertoryReceiptDataService.disapproveEnterReceipt(String receiptID) | | 审批不通过编号为receiptID的入库单 | |
| EnterRepertoryReceiptDataService.findEnterRepertoryByReceiptID (String receiptID) | | 根据输入的入库单编号，返回对应的EnterRepertoryByReceiptPO对象 | |
| EnterRepertoryReceiptDataService.findEnterReceiptByCreatorID(String creatorID) | | 根据输入的制单人编号，返回该员工制定的EnterRepertoryByReceiptPO对象列表 | |
| EnterRepertoryReceiptDataService.findEnterReceiptByCreatorAndKeyword(String creator, String keyword) | | 根据输入的制单人编号和关键字，返回该员工制定的、包含关键字keyword的EnterRepertoryByReceiptPO对象列表 | |
| EnterRepertoryReceiptDataService.getAllSubmitedEnterReceipts() | | 返回所有待审批的入库单列表 | |
| EnterRepertoryReceiptDataService.getAllEnterReceipts() | | 返回所有的入库单列表 | |
| LeaveRepertoryReceiptDataService.addLeaveRepertoryReceipt(LeaveRepertoryReceiptPO errpo) | | 插入单一持久化对象LeaveRepertoryReceiptPO对象 | |
| LeaveRepertoryReceiptDataService.sendLeaveReceipt(String receiptID) | | 将编号为receiptID的出库单提交至总经理审批 | |
| LeaveRepertoryReceiptDataService.approveLeaveReceipt(String receiptID) | | 审批通过编号为receiptID的出库单 | |
| LeaveRepertoryReceiptDataService.disapproveLeaveReceipt(String receiptID) | | 审批不通过编号为receiptID的出库单 | |
| LeaveRepertoryReceiptDataService.findLeaveRepertoryByReceiptID (String receiptID) | | 根据输入的出库单编号，返回对应的LeaveRepertoryByReceiptPO对象 | |
| LeaveRepertoryReceiptDataService.findLeaveReceiptByCreatorID(String creatorID) | | 根据输入的制单人编号，返回该员工制定的LeaveRepertoryByReceiptPO对象列表 | |
| LeaveRepertoryReceiptDataService.findLeaveReceiptByCreatorAndKeyword(String creator, String keyword) | | 根据输入的制单人编号和关键字，返回该员工制定的、包含关键字keyword的LeaveRepertoryByReceiptPO对象列表 | |
| LeaveRepertoryReceiptDataService.getAllSubmitedLeaveReceipts() | | 返回所有待审批的出库单列表 | |
| LeaveRepertoryReceiptDataService.getAllLeaveReceipts() | | 返回所有的出库单列表 | |

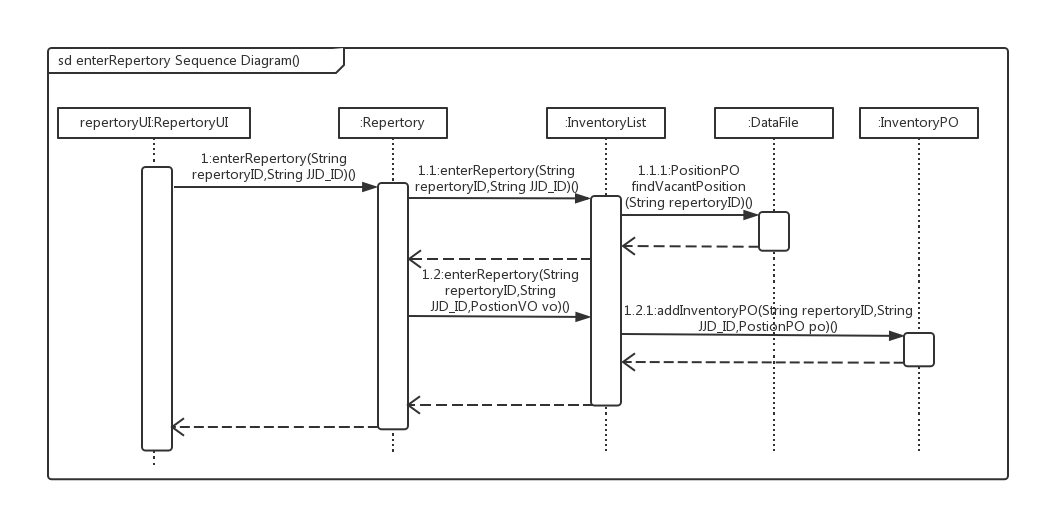
#### 4.5.2.4 业务逻辑层的动态模型

图5-2表明了快递物流系统中，当仓库管理员开始库存信息初始化任务后，仓库管理业务逻辑处理的相关对象的协作。



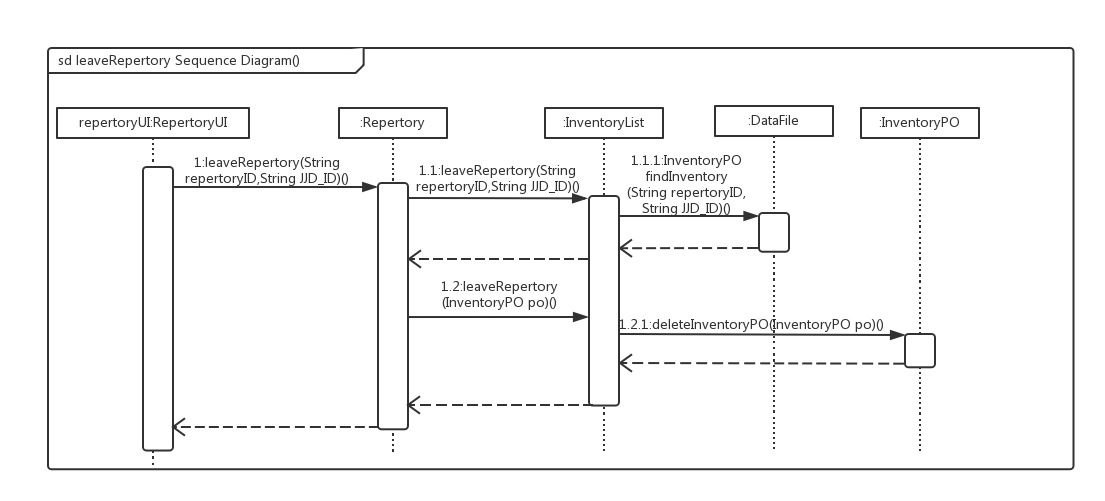
**图5-2 库存信息初始化的顺序图**

图5-3表明了快递物流系统中，当仓库管理员开始入库任务后，仓库管理业务逻辑处理的相关对象的协作。



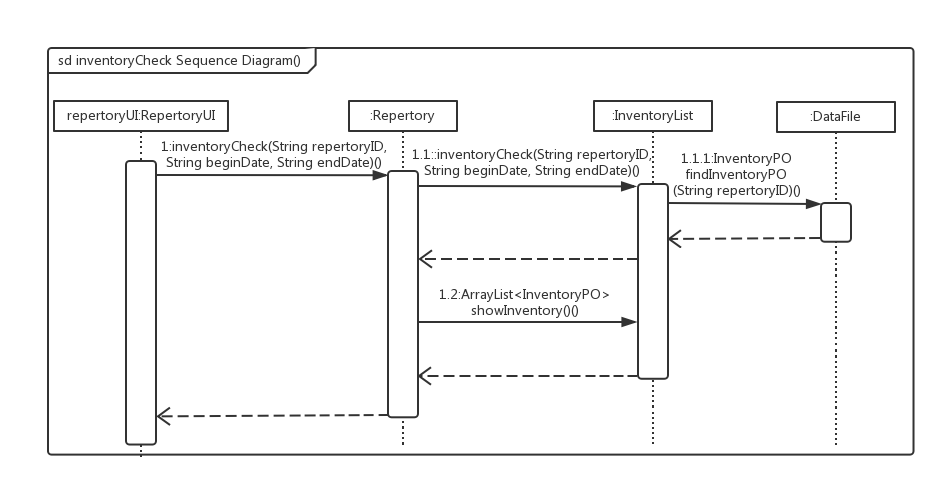
**图5-3 入库的顺序图**

图5-4表明了快递物流系统中，当仓库管理员开始出库任务后，仓库管理业务逻辑处理的相关对象的协作。



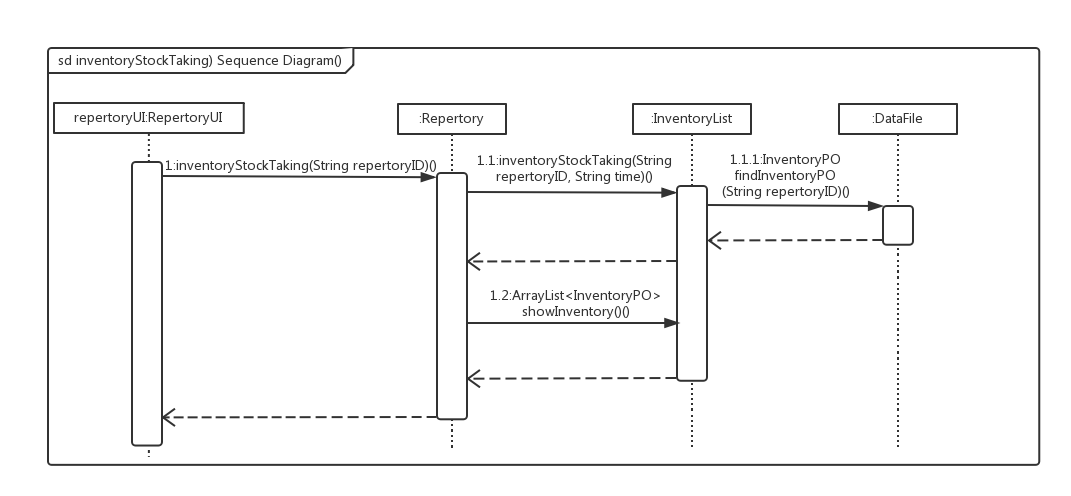
**图5-4 出库的顺序图**

图5-5表明了快递物流系统中，当仓库管理员开始库存查看任务后，仓库管理业务逻辑处理的相关对象的协作。



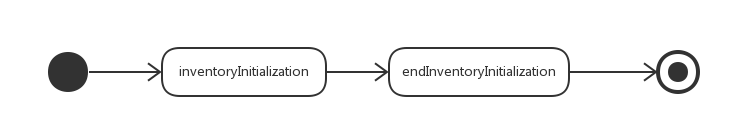
**图5-5 库存查看的顺序图**

图5-6表明了快递物流系统中，当仓库管理员开始库存盘点任务后，仓库管理业务逻辑处理的相关对象的协作。



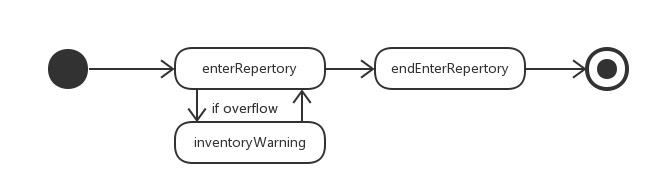
**图5-6 库存盘点的顺序图**

如图5-7所示的状态图描述了RepertoryBL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inventoryInitialization方法被UI调用，RepertoryBL进入inventoryInitialization状态；仓库管理员输入最大容量和警戒比例后，结束库存信息初始化进入endInventoryInitialization状态。



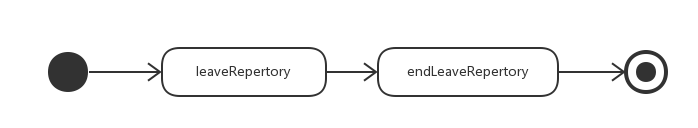
**图5-7 库存信息初始化的状态图**

如图5-8所示的状态图描述了RepertoryBL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enterRepertory方法被UI调用，RepertoryBL进入enterRepertory状态；当仓库管理员选择分区后，系统根据被选择分区是否超过警戒比例进入inventoryWarning状态要求再次确认入库，或者endEnterRepertory状态完成入库操作。



**图5-8 入库的状态图**

如图5-9所示的状态图描述了RepertoryBL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leaveRepertory方法被UI调用，RepertoryBL进入leaveRepertory状态；仓库管理员确认后，最后进入endEnterRepertory状态完成出库操作。



**图5-9 出库的状态图**

如图5-10所示的状态图描述了RepertoryBL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inventoryCheck方法被UI调用，RepertoryBL进入inventoryCheck状态；最后通过endInventoryCheck方法确定查看结束，进入endShow状态。



**图5-10 库存查看的状态图**

如图5-11所示的状态图描述了RepertoryBL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inventoryStockTaking方法被UI调用，RepertoryBL进入inventoryStockTaking状态；最后通过endInventoryStockTaking方法确定查看结束，进入endShow状态。



**图5-11 库存盘点的状态图**

### 4.2.6financebl模块

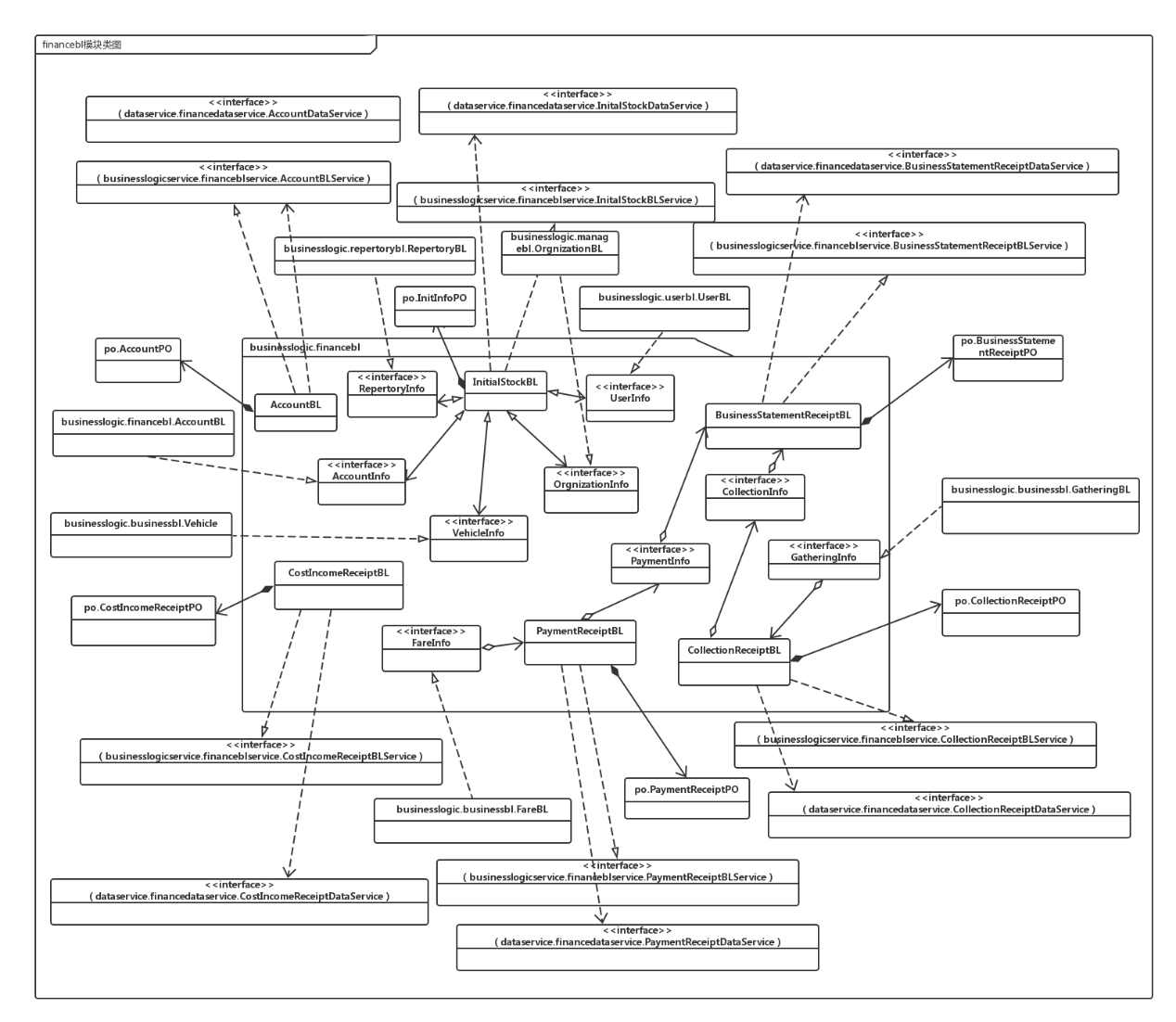
#### 4.2.6.1模块概述

financebl模块承担的需求参见ELS需求规格说明文档功能需求及相关非功能性需求。

financebl模块的职责及接口参见ELS体系结构设计文档。

#### 4.2.6.2整体结构

根据体系结构的设计，我们将系统分为展示层、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。比如展示层和业务逻辑层之间，我们添加businesslogicservice.financeblservice.CollectionReceiptBLService，businesslogicservice.financeblservice.PaymentReceiptBLService，businesslogicservice.financeblservice.CostIncomeReceiptBLService，businesslogicservice.financeblservice.BusinessStatementReceiptBLService，businesslogicservice.financeblservice.InitalStockBLService和businesslogicservice.financeblservice.AccountBLService接口。因为业务逻辑处理较少，根据专家模式没有添加controller类。业务逻辑层和数据层之间添加dataservice.financedataservice. CollectionReceiptDataService，dataservice.financedataservice. PaymentReceiptDataService，dataservice.financedataservice. CostIncomeReceiptDataService，dataservice.financedataservice. BusinessStatementReceiptDataService，dataservice.financedataservice. InitalStockDataService，dataservice.financedataservice. AccountDataService接口。CollectionReceiptPO,PaymentReceiptPO,BusinessStatementReceiptPO，CostIncomeReceiptPO，AccountPO和InitInfoPO是作为财务管理的持久化对象被添加到设计模型中去的。



**图6-1 financebl模块各个类的设计**

Financebl各个类的职责如表6-1所示

**表6-1 financebl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| CollectionReceiptBL | 入款单的领域模型对象，拥有一个入款单所持有的编号，操作员，入款金额等信息，可帮助完成结算管理界面所需要的服务。 |
| PaymentReceiptBL | 付款单的领域模型对象，拥有一个付款单所持有的编号，操作员，付款金额等信息，可帮助完成新建付款单界面所需要的服务。 |
| CostIncomeReceiptBL | 成本收益表的领域模型对象，拥有一个成本收益表所持有的编号，操作员，收入，支出金额等信息，可帮助完成新建成本收益表界面所需要的服务。 |
| BusinessStatementReceiptBL | 经营情况表的领域模型对象，拥有一个经营情况表所持有的入款记录，付款记录，编号等信息，可以帮助完成经营情况表界面所需要的服务。 |
| InitialStockBL | 期初建账的领域模型对象，拥有一个套账所持有的人员、机构、车辆、库存和账户等信息，可以帮助完成期初建账界面所需要的服务。 |
| AccountBL | 账户的领域模型对象，拥有账户的名称、金额等信息，可以帮助完成账户管理界面所需要的服务。 |

#### 4.2.6.3模块内部类的接口规范

CollectionReceiptBL，PaymentReceiptBL，CostIncomeReceiptBL，BusinessStatementReceiptBL，InitialStockBL和AccountBL的接口规范如表所示。

**表6-2 CollectionReceiptBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CollectionReceiptBL.  creatCollection | 语法 | | public int creatCollection (CollectionReceiptVO vo); |
| 前置条件 | | 用户要求新建入款单并输入有效的信息 |
| 后置条件 | | 向系统输入新的入款单信息 |
| CollectionReceiptBL.  getCollection | 语法 | | public CollectionReceiptVO getCollection (String s); |
| 前置条件 | | 用户具有查看入款单的权限 |
| 后置条件 | | 系统显示符合条件的入款单 |
| CollectionReceiptBL.  getAllCollection | 语法 | | public ArrayList<CollectionReceiptVO> getAllCollection(); |
| 前置条件 | | 用户有查看入款单的权限 |
| 后置条件 | | 系统显示所有入款单列表 |
| CollectionReceiptBL.  getGathering | 语法 | | public ArrayList<GatheringVO> getGathering(String HallID,String Time,GatheringVO vo); |
| 前置条件 | | 用户输入日期和营业厅编号并要求查看收款单 |
| 后置条件 | | 系统显示符合条件的收款单列表 |
| CollectionReceiptBL.  getMoney | 语法 | | public double[] getMoney(GatheringVO vo); |
| 前置条件 | | 所有符合条件的收款单已经获取 |
| 后置条件 | | 系统返回收款单的金额 |
| CollectionReceiptBL.  getTotalMoney | 语法 | | public double getTotalMoney(int[] money); |
| 前置条件 | | 用户要求合计金额 |
| 后置条件 | | 系统返回合计后的金额 |
| CollectionReceiptBL. getCollectionListID | 语法 | | public String getCollectionListID(); |
| 前置条件 | | 用户制定入款单时需要根据今天已经制定的入款单来自动生成入款单编号 |
| 后置条件 | | 系统显示自动生成的入款单ID |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FinanceDataService.creatCollection(CollectionReceiptPO po) | | 插入单一持久化对象CollectionReceiptPO | |
| FinanceDataService.getAllCollection() | | 得到持久化对象CollectionReceiptPO的ArrayList | |
| FinanceDataService.getGathering(String HallID,String Time) | | 得到持久化对象GatheringReceiptPO的ArrayList | |
| FinanceDataService.getMoney(GatheringReceiptPO po) | | 得到持久化对象GatheringReceiptPO中的金额数 | |
| FinanceDataService.getTotalMoney(int[] money) | | 得到持久化对象GatheringReceiptPO中的金额总和 | |
| FinanceDataService.getNum() | | 得到当天存储的持久化对象CollectionReceiptPO的个数 | |
| FinanceDataService.findByID(String ID) | | 根据ID返回对应的一个持久化对象CollectionReceiptPO | |
| FinanceDataService.modify(CollectionReceiptPO po) | | 修改单一持久化对象CollectionReceiptPO的信息 | |

**表6-3 PaymentReceiptBL的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PaymentReceiptBL.  createPayment | 语法 | public int creatPayment(PaymentReceiptVO vo); |
| 前置条件 | 用户要求新建付款单并输入有效信息 |
| 后置条件 | 向系统输入新的付款单信息 |
| PaymentReceiptBL.  getPayment | 语法 | Public PaymentReceiptVO getPayment(String s); |
| 前置条件 | 用户有查看付款单的权限 |
| 后置条件 | 系统显示符合条件的付款单列表 |
| PaymentReceiptBL.  getAllPayment | 语法 | public ArrayList<PaymentReceiptVO> getAllPayment(); |
| 前置条件 | 用户有查看付款单的权限 |
| 后置条件 | 系统显示所有付款单列表 |
| PaymentReceiptBL.  getPaymentID | 语法 | public String getPaymentListID(); |
| 前置条件 | 用户制定付款单时需要根据今天已经制定的付款单来自动生成付款单编号 |
| 后置条件 | 系统显示自动生成的付款单ID |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| FinanceDataService. creatPaymentReceipt  (PaymentReceiptPO po) | | 插入单一持久化对象 PaymentReceiptPO |
| FinanceDataService. getAllPaymentReceipt() | | 得到持久化对象 PaymentReceiptPO的ArrayList |
| FinanceDataService.  getNum() | | 得到当天存储的持久化对象PaymentReceiptPO的个数 |
| FinanceDataService. findByID(String ID) | | 根据ID返回对应的一个持久化对象PaymentReceiptPO |
| FinanceDataService. modify(PaymentReceiptPO po) | | 修改单一持久化对象PaymentReceiptPO的信息 |

**表6-4 CostIncomeReceiptBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CostIncomeReceiptBL.  createCostIncomeList | 语法 | | public int creatCostIncomeList(CostIncomeReceiptVO vo); |
| 前置条件 | | 用户要求新建成本收益表并输入有效信息 |
| 后置条件 | | 向系统中输入新的成本收益表信息 |
| CostIncomeReceiptBL.  getCostIncmeList | 语法 | | public CostIncomeReceiptVO getCostIncomeList(String s); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回符合条件的成本收益表 |
| CostIncomeReceiptBL.  getAllCostIncomeList | 语法 | | public ArrayList<CostIncomeReceiptVO> getAllCostIncomeList(); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回所有成本收益表列表 |
| CostIncomeReceiptBL.  getCollection | 语法 | | public ArrayList<CollectionReceiptVO> getCollection(); |
| 前置条件 | | 用户有查看入款单的权限 |
| 后置条件 | | 系统返回所有入款单列表 |
| CostIncomeReceiptBL.  getPayment | 语法 | | public ArrayList<PaymentReceiptVO>getPayment(); |
| 前置条件 | | 用户有查看付款单的权限 |
| 后置条件 | | 系统返回所有付款单列表 |
| CostIncomeReceiptBL.  getCost | 语法 | | public double getCost(PaymentReceiptVO vo); |
| 前置条件 | | 已获取付款单 |
| 后置条件 | | 系统返回付款单金额总和 |
| CostIncomeReceiptBL.  getIncome | 语法 | | public double getIncome(CollectionReceiptVO vo); |
| 前置条件 | | 已获取入款单 |
| 后置条件 | | 系统返回入款单金额总和 |
| CostIncomeReceiptBL.  getProfit | 语法 | | public double getProfit(double income,double cost); |
| 前置条件 | | 已获取付款单金额之和,入款单金额之和 |
| 后置条件 | | 系统返回利润（即入款单付款单金额之差） |
| CostIncomeReceiptBL.  getCostIncomeListID | 语法 | | public String getCostIncomeListID(); |
| 前置条件 | | 用户制定成本收益表时需要根据今天已经制定的成本收益表来自动生成编号 |
| 后置条件 | | 系统显示自动生成的成本收益表ID |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FinanceDataService.creatCostIncomeList(CostIncomeReceiptPO po) | | 插入单一持久化对象 CostIncomeReceiptPO | |
| FinanceDataService. getAllCostIncomeList() | | 得到持久化对象CostIncomeReceiptPO的ArrayList | |
| FinanceDataService. getCollection() | | 得到持久化对象CollectionReceiptPO的ArrayList | |
| FinanceDataService. getPayment() | | 得到持久化对象PaymentReceiptPO的ArrayList | |
| FinanceDataService. getNum() | | 得到当天存储的持久化对象CostIncomeReceiptPO的个数 | |
| FinanceDataService. findByID(String ID) | | 根据ID返回对应的一个持久化对象CostIncomeReceiptPO | |
| FinanceDataService. modify(CostIncomeReceiptPO po) | | 修改单一持久化对象CostIncomeReceiptPO的信息 | |

**表6-5 BusinessStatementReceiptBL的接口规范**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| BusinessStatementReceiptBL.showBSL | 语法 | | | public BusinessstatementReceiptVO show(String beginTime,String endTime); |
| 前置条件 | | | 用户输入起始日期并要求查看经营情况表 |
| 后置条件 | | | 系统显示该段时间内所有入款单和付款单信息 |
| BusinessStatementReceiptBL.showAllBSL | 语法 | | | public ArrayList<BusinessstatementReceiptVO> showAllBSList(); |
| 前置条件 | | | 用户经验证有查看经营情况表的权限 |
| 后置条件 | | | 系统显示所有经营情况的列表 |
| BusinessStatementReceiptBL.export | 语法 | | | public int export(BusinessstatementReceiptVO vo); |
| 前置条件 | | | 用户要求导出经营情况表的excel |
| 后置条件 | | | 系统生成经营情况表的excel |
| 需要的服务（需接口） | | | | |
| 服务名 | | | 服务 | |
| FinanceDataService.showBSL(String beginTime,String endTime) | | 返回并输出符合时间条件的单一持久化对象BusinessstatementReceiptPO | | |
| public ArrayList<BusinessstatementReceiptPO> showAllBSList() | | 返回并输出所有的单一持久化对象BusinessstatementReceiptPO | | |
| public int export(BusinessstatementReceiptPO po) | | 导出单一持久化对象BusinessstatementReceiptPO到excel | | |

**表6-6 InitialStockBL的接口规范**

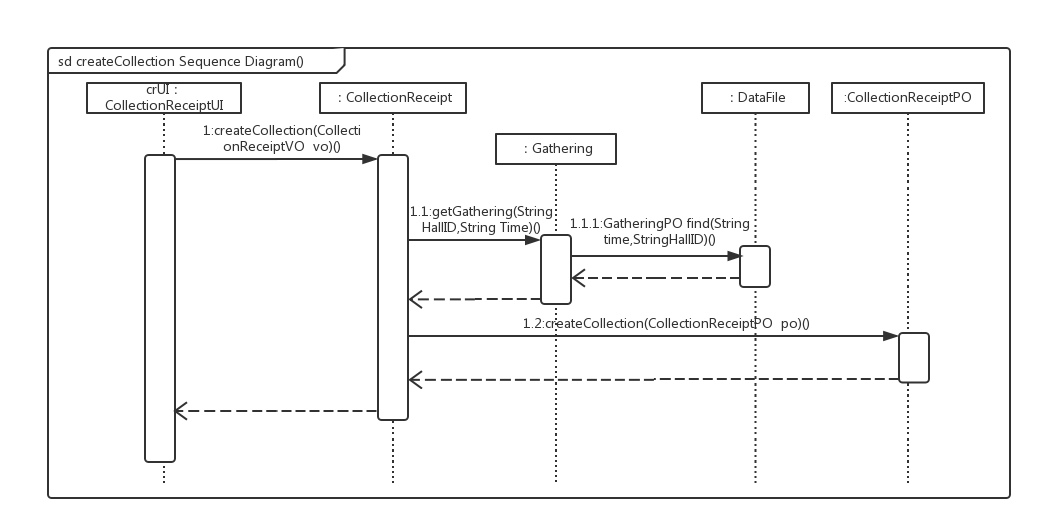
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InitialStockBL.  initInfo | 语法 | public int initInfo(InitInfoVO vo,String time); |
| 前置条件 | 用户要求期初建账 |
| 后置条件 | 系统初始化一套账 |
| InitialStockBL.  getInitInfo | 语法 | public InitInfoVO getInitInfo(String time); |
| 前置条件 | 用户要求查看期初信息 |
| 后置条件 | 返回期初信息并显示 |
| InitialStockBL.  getAllInitInfo | 语法 | public ArrayList<InitInfoVO> getAllInitInfo(); |
| 前置条件 | 用户具有查看期初信息的权限 |
| 后置条件 | 系统返回所有期初信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| FinanceDataService.initInfo(InitInfoPO po) | 在数据库中添加一套账的初始化信息（InitInfoPO） | |
| FinanceDataService.getInitInfo(String time) | 查找符合时间条件的期初信息并返回 | |
| FinanceDataService.getAllInitInfo() | 返回并输出所有的持久化对象InitInfoPO | |

**图6-7 AccountBL的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Account.  addAccount | 语法 | public int addAccount(AccountVO vo); | |
| 前置条件 | 用户要求新建账户 | |
| 后置条件 | 在账户列表中添加该账户信息 | |
| Account.  deleteAccount | 语法 | public int deleteAccount(AccountVO vo); | |
| 前置条件 | 用户要求删除账户 | |
| 后置条件 | 从账户列表中删除该账户信息 | |
| Account.  modifyAccount | 语法 | public int modifyAccount(AccountVO vo,String name) | |
| 前置条件 | 用户要求修改账户并选中需修改的账户 | |
| 后置条件 | 在账户列表中修改该账户信息 | |
| Account.  findByName | 语法 | public AccountVO findByName(String name); | |
| 前置条件 | 用户输入账户名称并查找 | |
| 后置条件 | 查找是否存在该名称的账户并返回该账户的信息 | |
| Account.  findByKeyword | 语法 | public ArrayList<AccountVO> findByKeyword(String s); | |
| 前置条件 | 用户输入关键词并查询 | |
| 后置条件 | 查找是否存在含有该关键字的账户并返回含有该关键字的账户的信息 | |
| Account.  showAll | 语法 | public ArrayList<AccountVO> showAll(); | |
| 前置条件 | 用户要求查看所有的账户列表 | |
| 后置条件 | 返回所有账户列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DataFactory.getAccountData | | | 得到Account数据库的引用 |
| FinanceDataService.addAccount(AccountPO po) | | | 插入单一持久化对象AccountPO |
| FinanceDataService.deleteAccount(AccountPO po) | | | 删除单一持久化对象AccountPO |
| FinanceDataService.modifyAccount(AccountPO po,String name) | | | 修改单一持久化对象AccountPO |
| FinanceDataService.findByName(String name) | | | 根据输入的名称查找单一持久化对象AccountPO |
| public ArrayList<AccountPO> findByKeyword(String s) | | | 根据输入的关键字查找持久化对象AccountPO |
| public ArrayList<AccountPO> showAll() | | | 输出持久化对象AccountPO |

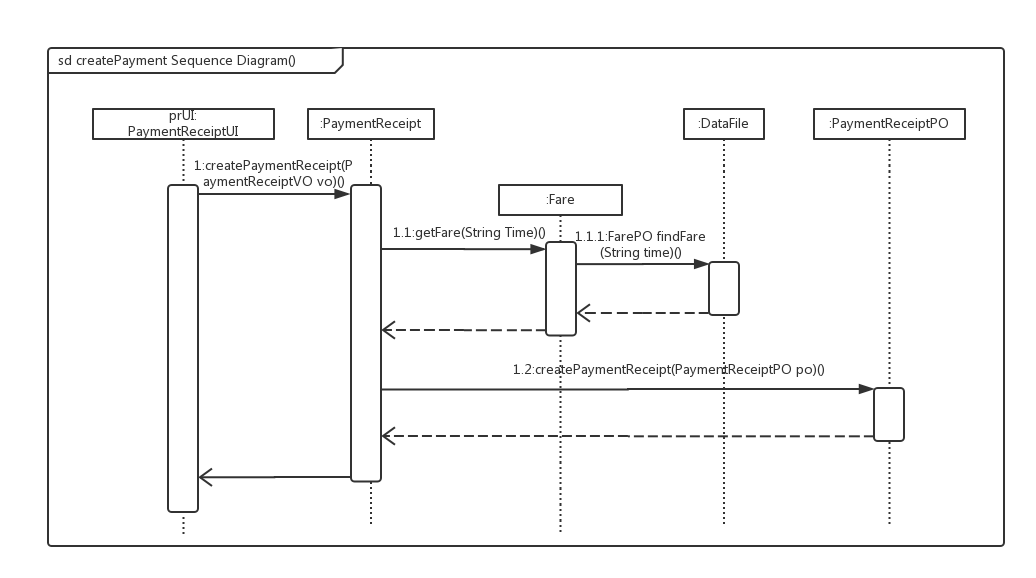
#### 4.2.6.4业务逻辑层的动态模型

图6-2表示了快递物流系统中，当财务人员进行结算管理时，业务逻辑层处理的相关对象之间的协作。



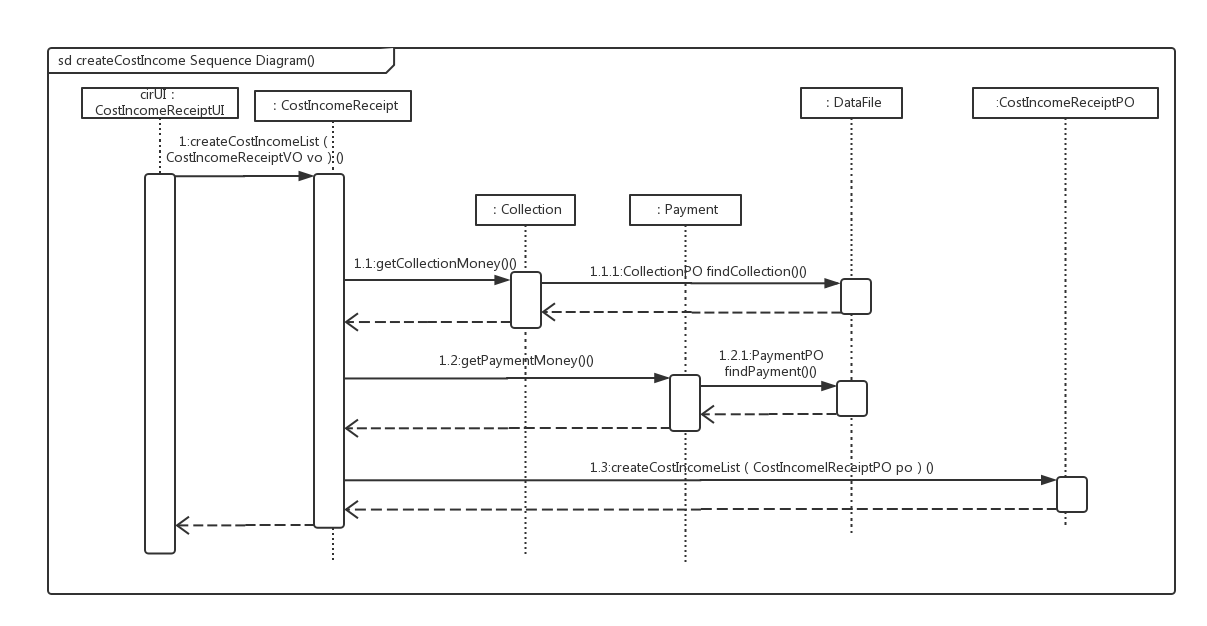
**图6-2 结算管理的顺序图**

图6-3表示了快递物流系统中，当财务人员进行新建付款单时，业务逻辑层处理的相关对象之间的协作。



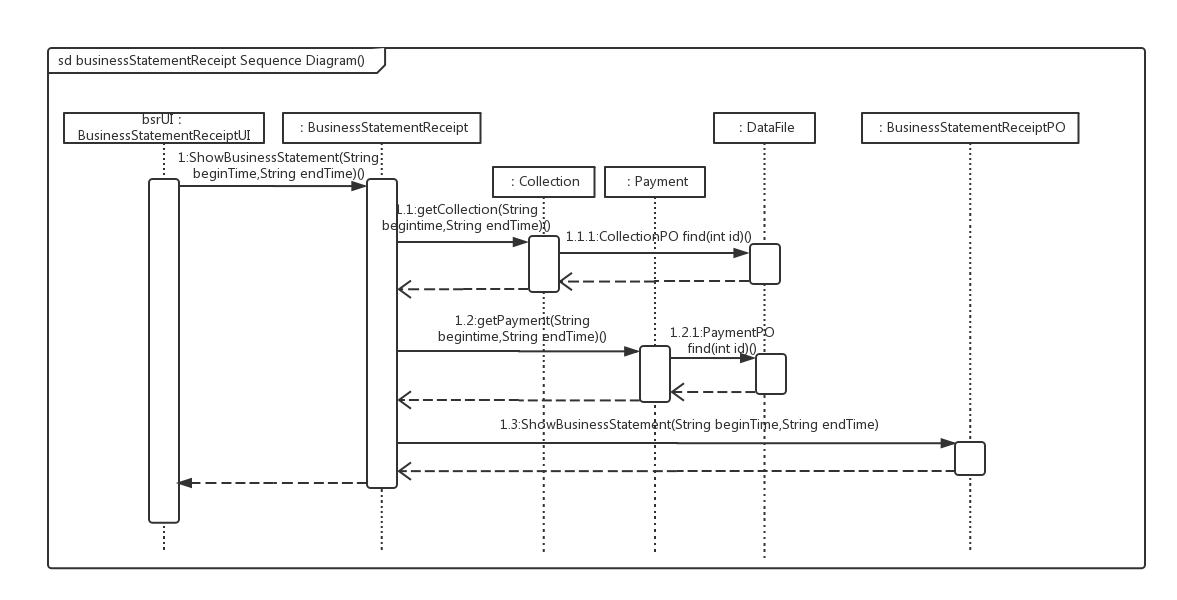
**图6-3 新建付款单的顺序图**

图6-4表示了快递物流系统中，当财务人员进行新建成本收益表时，业务逻辑层处理的相关对象之间的协作。



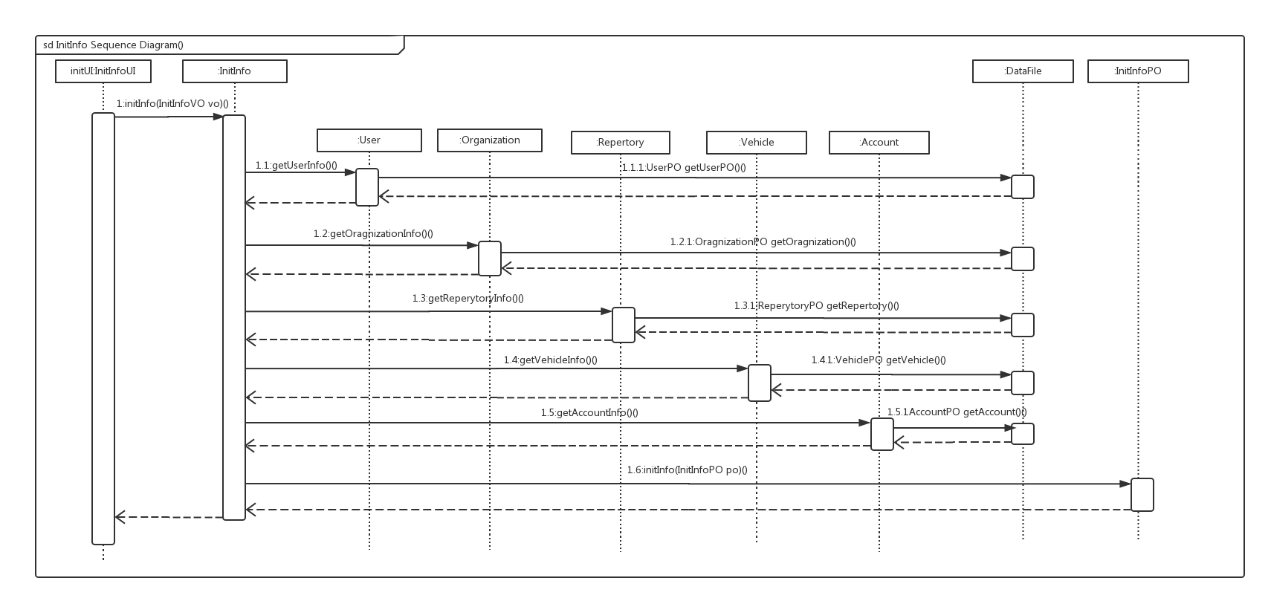
**图6-4 新建成本收益表的顺序图**

图6-5表示了快递物流系统中，当财务人员进行查看经营情况表时，业务逻辑层处理的相关对象之间的协作。



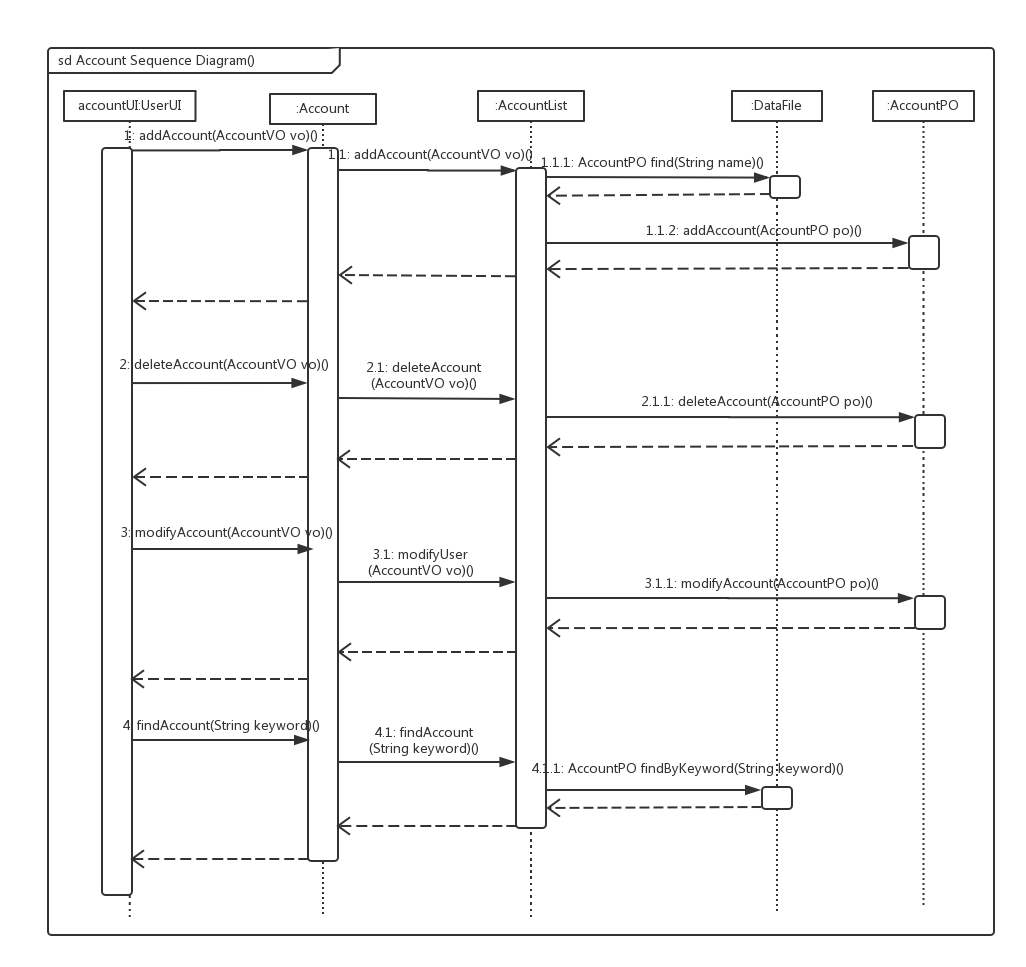
**图6-5 查看经营情况表的顺序图**

图6-6表示了快递物流系统中，当财务人员进行期初建账时，业务逻辑层处理的相关对象之间的协作。



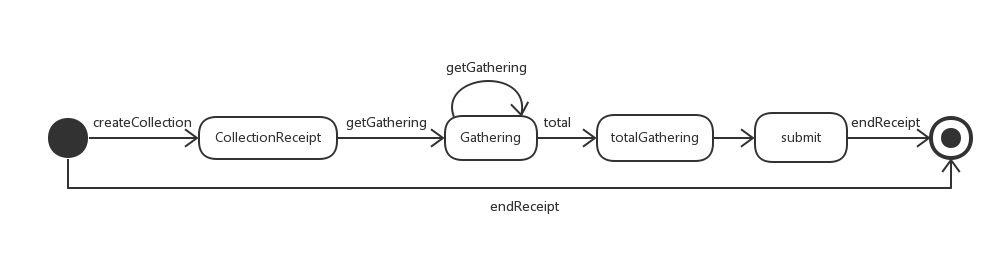
**图6-6 期初建账的顺序图**

图6-7表示了快递物流系统中，当财务人员进行账户管理时，业务逻辑层处理的相关对象之间的协作。



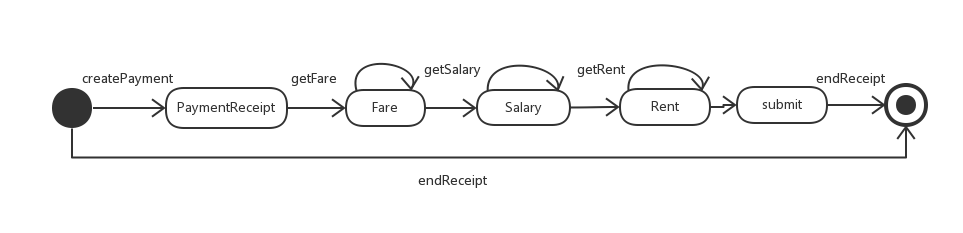
**图6-7 账户管理的顺序图**

如图6-8所示，CollectionReceipt对象状态图描述了CollectionReceipt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createCollection方法被 UI调用，进入CollectionReceipt状态；之后通过获取收款单进入Gathering状态。随后，通过计算收款单金额之和进入totalGathering状态；随着submit方法被UI调用，CollectionReceipt进入endReceipt状态。



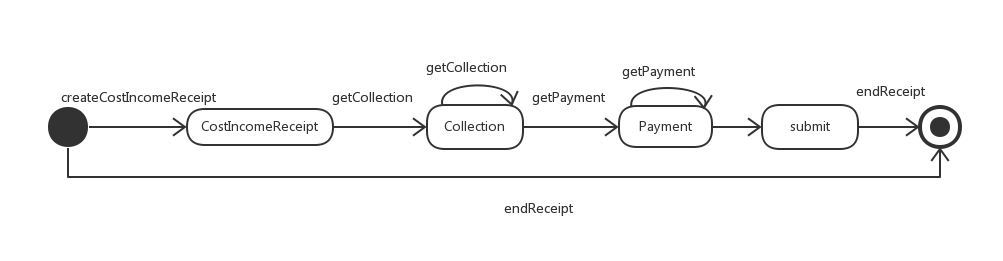
**图6-8 结算管理的状态图**

如图6-9所示，PaymentReceipt对象状态图描述了PaymentReceipt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createPayment方法被 UI调用，进入PaymentReceipt状态；之后通过获取运费单进入Fare状态，通过输入薪水进入Salary状态，通过输入租金进入Rent状态；随着submit方法被UI调用，PaymentReceipt进入endReceipt状态。



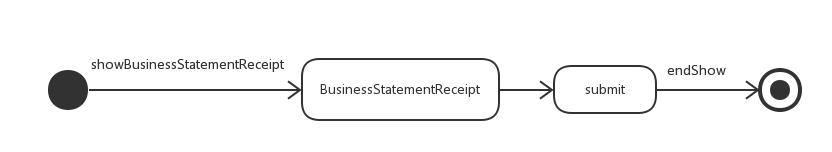
**图6-9 新建付款单的状态图**

如图6-10所示，CostIncomeReceipt对象状态图描述了CostIncomeReceipt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createCostIncome方法被 UI调用，进入CostIncomeReceipt状态；之后通过获取入款单进入Collection状态，通过获取付款单进入Payment状态；随着submit方法被UI调用，CostIncomeReceipt进入endReceipt状态。



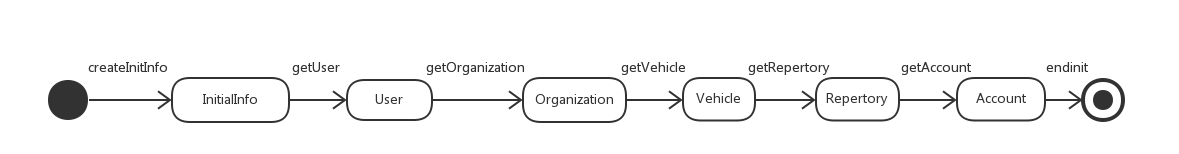
**图6-10 新建成本收益表的状态图**

如图6-11所示，BusinessStatementReceipt对象状态图描述了BusinessStatementReceipt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showBusinessStatementReceipt被 UI调用，进入BusinessStatementReceipt状态；随着submit方法被UI调用，BusinessStatementReceipt进入endShow状态。



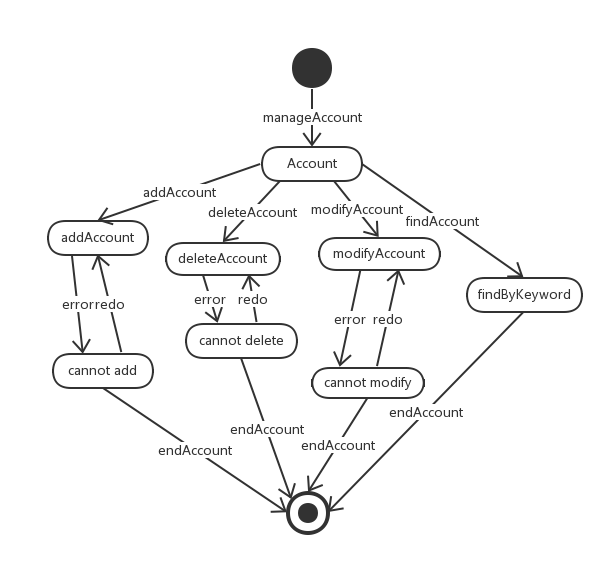
**图6-11 查看经营情况表的状态图**

如图6-12所示, InitInfo对象状态图描述了InitInfo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createInitInfo方法被 UI调用，进入InitialInfo状态；之后通过添加用户信息进入User状态，通过添加机构信息进入Organization状态，通过添加车辆信息进入Vehicle状态，通过添加库存信息进入Repertory状态，通过添加账户信息进入Account状态；最后进入endInit状态。



**图6-12 期初建账的状态图**

如图6-13所示，Account对象状态图描述了Account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manageAccount方法被 UI调用，Account进入Account状态；之后通过添加账户进入addAccount状态，通过删除账户进入deleteAccount状态，通过修改账户进入modifyAccount状态，通过查询账户进入findByKeyword状态；最后进入endAccount状态。



**图6-13 账户管理的状态图**

### 4.2.7managebl模块

#### 4.2.7.1模块概述

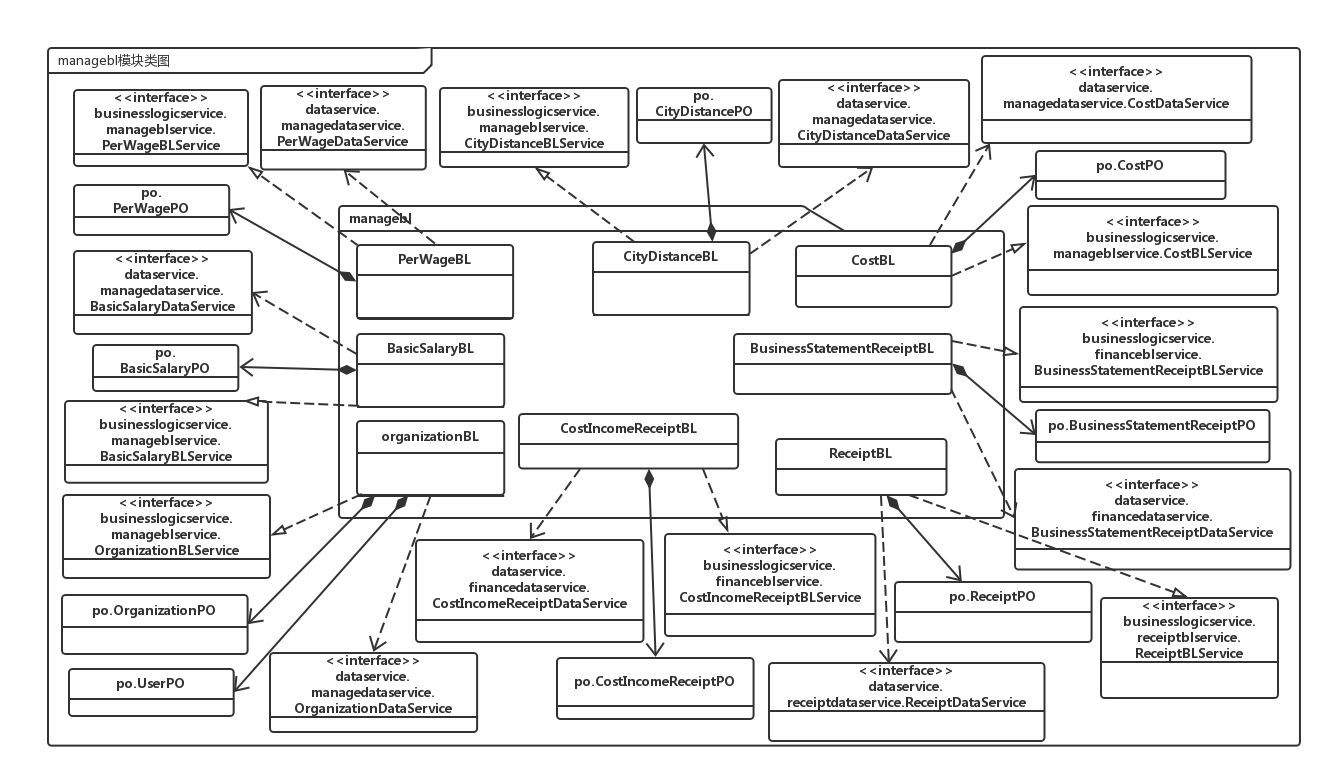
managebl模块承担的需求参见ELS需求规格说明文档功能需求及相关非功能需求。

managebl模块的职责及接口参见ELS体系结构设计文档表14。

#### 4.2.7.2整体结构

根据体系结构的设计，我们将系统分为展示层、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。比如展示层和业务逻辑层之间，我们添加businesslogicservice.manageblservice.OrganizationBLService、businesslogicservice.manageblservice.BasicSalaryBLService、businesslogicservice.manageblservice.CityDistanceBLService等接口。业务逻辑层和数据层之间添加dataservice.managedataservice.OrganizationDataService、dataservice.managedataservice.BasicSalaryDataService、ataservice.managedataservice.CityDistanceDataService等接口。因为业务逻辑处理较少，根据专家模式没有添加Controller类。OrganizationPO等PO对象是作为总经理管理记录的持久化对象被添加到设计模型中去的。

managebl模块的设计如图7-1所示。



**图7-1 managebl模块各个类的设计**

managebl模块各个类的职责如表7-1所示。

**表7-1 managebl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrganizationBL | 机构的领域模型对象，拥有机构数据的类别、编号、名称、下属仓库（若是营业厅则无仓库），可以解决总经理机构管理的问题。 |
| ReviewReceiptBL | 单据审批的领域模型对象，可以解决总经理单据审批的问题。 |
| BusinessStatementReceiptBL | 经营情况表的领域模型对象，拥有一个经营情况表所持有的入款记录、付款记录、编号等信息，可以帮助完成查看经营情况表界面所需要的服务。 |
| CostIncomeReceiptBL | 成本收益表的领域模型对象，拥有一个经营情况表所持有的编号、操作员、收入、支出金额等信息，可以帮助完成查看成本收益表界面所需要的服务。 |
| PerWageBL | 每次工资的领域模型对象，拥有每次工资所持有的职业、工资金额等信息，可以解决总经理每次工资管理的问题。 |
| BasicSalaryBL | 基础月薪的领域模型对象，拥有基础月薪所持有的职业、月薪金额等信息，可以解决总经理基础月薪管理的问题。 |
| CityDistanceBL | 城市距离的领域模型对象，拥有城市距离所持有的城市、距离数等信息，可以解决总经理城市距离管理的问题。 |
| CostBL | 运费系数的领域模型对象，拥有运费系数所持有的类型、运费系数等信息，可以解决总经理运费系数管理的问题。 |

#### 4.2.7.3模块内部类的接口规范

OrganizationBL的接口规范如表7-2所示。

**表7-2 OrganizationBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrganizationBL.addOrganization | 语法 | | public int addOrganization(OrganizationVO organizationvo) |
| 前置条件 | | 经过验证的总经理有添加新机构的权利，且总经理输入了完整有效的新机构信息 |
| 后置条件 | | 系统保存新增的机构信息并更新机构列表 |
| OrganizationBL.deleteOrganization | 语法 | | public int deleteOrganization(OrganizationVO organizationvo) |
| 前置条件 | | 经过验证的总经理有删除机构的权利，且总经理输入了有效的待删除机构信息 |
| 后置条件 | | 系统查找对应的机构信息，并删除该机构对象 |
| OrganizationBL.modifyOrganization | 语法 | | public int modifyOrganization(OrganizationVO organizationvo) |
| 前置条件 | | 经过验证的总经理有修改机构信息的权利，且总经理输入了有效的待修改机构信息 |
| 后置条件 | | 系统更新机构信息 |
| OrganizationBL.findOrganization | 语法 | | public OrganizationVO findOrganization(String organizationID) |
| 前置条件 | | 经过验证的总经理有查询机构的权利，且总经理输入了有效的待查询机构编号 |
| 后置条件 | | 系统根据输入的机构编号查找机构，并返回机构信息 |
| OrganizationBL.showAllOrganizations() | 语法 | | public ArrayList<OrganizationVO> showAllOrganizations() |
| 前置条件 | | 经过验证的总经理有查看全部机构的权利 |
| 后置条件 | | 系统返回全部的机构信息 |
| OrganizationBL.chooseDepartment | 语法 | | public int chooseDepartment(String userID, String organizationID) |
| 前置条件 | | 经过验证的总经理有为新员工选择所属机构的权利，且总经理输入了有效的员工编号和机构编号 |
| 后置条件 | | 系统根据输入的员工编号查找对应的员工，并将该员工所属机构修改为机构编号对应的机构 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ManageDataService.OrganizationDataService.addOrganization(OrganizationPO organizationpo) | | 在管理数据包的机构数据文件中插入一个OrganizationPO对象 | |
| ManageDataService.OrganizationDataService.deleteOrganization(OrganizationPO organizationpo) | | 在管理数据包的机构数据文件中删除一个OrganizationPO对象 | |
| ManageDataService.OrganizationDataService.modifyOrganization(OrganizationPO organizationpo) | | 在管理数据包的机构数据文件中更新一个OrganizationPO对象 | |
| ManageDataService.OrganizationDataService.findOrganization(String OrganizationID) | | 根据输入的机构编号，返回对应的OrganizationPO对象 | |
| ManageDataService.OrganizationDataService.OrganizationDataService.showAllOrganizations | | 返回所有的OrganizationPO对象列表 | |
| ManageDataService.OrganizationDataService.modifyUser(UserPO userpo) | | 在用户数据文件中更新一个UserPO对象 | |

ReviewReceiptBL的接口规范如表7-3所示。

**表7-3 ReviewReceiptBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| ReviewReceipt.approve | 语法 | public int approve(String receiptID, Object ob) | |
| 前置条件 | 总经理选中了有效单据并确认审批通过 | |
| 后置条件 | 将该单据状态设为通过并更新单据信息 | |
| ReviewReceipt.getAllReceiptList() | 语法 | public AllReceiptShowVO getAllReceiptList() | |
| 前置条件 | 已有单据记录 | |
| 后置条件 | 返回所有单据信息 | |
| ReviewReceipt.refresh | 语法 | public AllReceiptShowVO refresh() | |
| 前置条件 | 已有单据记录 | |
| 后置条件 | 刷新所有待审批单据信息 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 |
| FileFactory.getBusinessData | | | 得到Business文件的引用 |
| FileFactory.getIntermediateData | | | 得到Intermediate文件的引用 |
| FileFactory.getCollectionReceiptData | | | 得到CollectionReceipt文件的引用 |
| FileFactory.getPaymentReceiptData | | | 得到PaymentReceipt文件的引用 |
| FileFactory.getEnterRepertoryReceiptData | | | 得到EnterRepertoryReceipt文件的引用 |
| FileFactory.getLeaveRepertoryReceiptData | | | 得到LeaveRepertoryReceipt文件的引用 |
| BusinessDataService.getSubmittedGatheringReceiptInfo() | | | 返回全部提交的收款单 |
| BusinessDataService.saveGatheringReceiptInfo(GatheringReceiptPO grPO) | | | 在管理数据包的营业厅数据文件中更新一个GatheringReceiptPO对象 |
| BusinessDataService.getSubmittedEnVehicleReceiptInfo() | | | 返回全部提交的营业厅装车单 |
| BusinessDataService.saveEnVehicleReceiptInfo(EnVehicleReceiptPO evrPO) | | | 在管理数据包的营业厅数据文件中更新一个EnVehicleReceiptPO对象 |
| BusinessDataService.getSubmittedOrderAcceptReceiptInfo() | | | 返回全部提交的营业厅到达单 |
| BusinessDataService.saveOrderAcceptReceiptInfo(OrderAcceptReceipt oarPO oarPO) | | | 在管理数据包的营业厅数据文件中更新一个OrderAcceptReceipt oarPO对象 |
| BusinessDataService.getSubmittedDistributeReceiptInfo() | | | 返回全部提交的派件单 |
| BusinessDataService.saveDistributeReceiptInfo(DistributeReceiptPO drPO) | | | 在管理数据包的营业厅数据文件中更新一个DistributeReceiptPO对象 |
| IntermediateDataService.getSubmittedEnIntermediateReceiptInfo() | | | 返回全部提交的中转中心装车单 |
| IntermediateDataService.saveEnIntermediateReceiptInfo(EnIntermediateReceiptPO erPO) | | | 在管理数据包的中转中心数据文件中更新一个EnIntermediateReceiptPO对象 |
| IntermediateDataService.getSubmittedTransferingReceiptInfo() | | | 返回全部提交的中转中心到达单 |
| IntermediateDataService.saveTransferingReceiptInfo(TransferingReceiptPO trPO) | | | 在管理数据包的中转中心数据文件中更新一个TransferingReceiptPO对象 |
| CollectionReceiptDataService.getUnapprovedCollectionReceipt() | | | 返回全部提交的合计收款单 |
| CollectionReceiptDataService. saveSubmittedCollectionReceiptInfo(CollectionReceiptPO crPO) | | | 在管理数据包的财务人员数据文件中更新一个CollectionReceiptPO对象 |
| paymentReceiptDataService.getUnapprovedPaymentReceipt() | | | 返回全部提交的付款单 |
| paymentReceiptDataService.saveSubmittedPaymentReceiptInfo(PaymentReceiptPO prPO) | | | 在管理数据包的财务人员数据文件中更新一个PaymentReceiptPO对象 |
| RepertoryDataService.EnterRepertoryReceiptDataService.getAllSubmitedEnterRepertoryReceipts() | | | 返回全部提交的入库单 |
| RepertoryDataService.EnterRepertoryReceiptDataService.approveEnterRepertoryReceipt(String receiptID) | | | 在管理数据包的仓库管理员数据文件中更新一个EnterRepertoryReceiptPO对象 |
| RepertoryDataService.LeaveRepertoryReceiptDataService.getAllSubmitedLeaveRepertoryReceipts() | | | 返回全部提交的出库单 |
| RepertoryDataService.LeaveRepertoryReceiptDataService.approveLeaveRepertoryReceipt(String receiptID) | | | 在管理数据包的仓库管理员数据文件中更新一个LeaveRepertoryReceiptPO对象 |

BusinessStatementReceiptBL的接口规范如表7-4所示。

**表7-4 BusinessStatementReceiptBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| BusinessStatementReceiptBL.showBSL | 语法 | | public BusinessstatementReceiptVO show(String beginTime,String endTime); |
| 前置条件 | | 用户输入起始日期并要求查看经营情况表 |
| 后置条件 | | 系统显示该段时间内所有入款单和付款单信息 |
| BusinessStatementReceiptBL.showAllBSL | 语法 | | public ArrayList<BusinessstatementReceiptVO> showAllBSList(); |
| 前置条件 | | 用户经验证有查看经营情况表的权限 |
| 后置条件 | | 系统显示所有经营情况的列表 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FinanceDataService.showBSL(String beginTime,String endTime) | | 返回并输出符合时间条件的单一持久化对象BusinessstatementReceiptPO | |
| public ArrayList<BusinessstatementReceiptPO> showAllBSList() | | 返回并输出所有的单一持久化对象BusinessstatementReceiptPO | |

CostIncomeReceiptBL的接口规范如表7-5所示。

**表7-5 CostIncomeReceiptBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CostIncomeReceiptBL.getCostIncmeList | 语法 | | public CostIncomeReceiptVO getCostIncomeList(String s); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回符合条件的成本收益表 |
| CostIncomeReceiptBL.getAllCostIncomeList | 语法 | | public ArrayList<CostIncomeReceiptVO> getAllCostIncomeList(); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回所有成本收益表列表 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FinanceDataService. getNum() | | 得到当天存储的持久化对象CostIncomeReceiptPO的个数 | |
| FinanceDataService. findByID(String ID) | | 根据ID返回对应的一个持久化对象CostIncomeReceiptPO | |

PerWageBL的接口规范如表7-6所示。

**表7-6 PerWageBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| PerWageBL.modifyPerWage | 语法 | | public int modifyPerWage(PerWageVO perwagevo) |
| 前置条件 | | 经过验证的总经理有修改每次工资的权利，且总经理输入了有效的待修改每次工资信息 |
| 后置条件 | | 系统更新每次工资信息 |
| PerWageBL.findPerWage | 语法 | | public double findPerWage(ProfessionType profession) |
| 前置条件 | | 经过验证的总经理有查询每次工资的权利，且总经理输入了有效的对应职业编号 |
| 后置条件 | | 系统根据输入的职业类型查找每次工资，并返回每次工资信息 |
| PerWageBL.showAllPerWages() | 语法 | | public ArrayList<PerWageVO> showAllPerWages() |
| 前置条件 | | 经过验证的总经理有查看全部每次工资信息的权利 |
| 后置条件 | | 系统返回全部的每次工资信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ManageDataService.PerWageDataService.modifyPerWage(PerWagePO perwagepo) | | 在管理数据包的每次工资数据文件中更新一个PerWagePO对象 | |
| ManageDataService.PerWageDataService.findPerWage(ProfessionType profession) | | 根据输入的职业类型，返回对应的PerWagePO对象 | |
| ManageDataService.PerWageDataService.showAllPerWages() | | 返回全部的PerWagePO对象 | |

BasicSalaryBL的接口规范如表7-7所示。

**表7-7 BasicSalaryBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| BasicSalaryBL.modifyBasicSalary | 语法 | | public int modifyBasicSalary(BasicSalaryVO basicsalaryvo) |
| 前置条件 | | 经过验证的总经理有修改基础月薪的权利，且总经理输入了有效的待修改基础月薪信息 |
| 后置条件 | | 系统更新基础月薪信息 |
| BasicSalaryBL.findBasicSalary | 语法 | | public double findBasicSalary(ProfessionType profession) |
| 前置条件 | | 经过验证的总经理有查询基础月薪的权利，且总经理输入了有效的对应职业编号 |
| 后置条件 | | 系统根据输入的职业类型查找基础月薪，并返回基础月薪信息 |
| BasicSalaryBL.showAllBasicSalarys | 语法 | | public ArrayList<BasicSalaryVO> showAllBasicSalarys() |
| 前置条件 | | 经过验证的总经理有查看全部每次基础月薪的权利 |
| 后置条件 | | 系统返回全部的基础月薪信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ManageDataService.BasicSalaryDataService.modifyBasicSalary(BasicSalaryPO citydistancepo) | | 在管理数据包的基础月薪数据文件中更新一个BasicSalaryPO对象 | |
| ManageDataService.BasicSalaryDataService.findBasicSalary(ProfessionType profession) | | 根据输入的职业类型，返回对应的BasicSalaryPO对象 | |
| ManageDataService.BasicSalaryDataService.showAllBasicSalarys() | | 返回全部的BasicSalaryPO对象 | |

CityDistanceBL的接口规范如表7-8所示。

**表7-8 CityDistanceBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CityDistanceBL.addCityDistance | 语法 | | public int addCityDistance(CityDistanceVO citydistancevo) |
| 前置条件 | | 经过验证的总经理有添加新的城市距离的权利，且总经理输入了完整有效的新城市距离信息 |
| 后置条件 | | 系统保存新增的城市距离信息并更新城市距离列表 |
| CityDistanceBL.deleteCityDistance | 语法 | | public int deleteCityDistance(CityDistanceVO citydistancevo) |
| 前置条件 | | 经过验证的总经理有删除城市距离的权利，且总经理输入了有效的待删除城市距离信息 |
| 后置条件 | | 系统查找对应的城市距离信息，并删除该城市距离对象 |
| CityDistanceBL.modifyCityDistance | 语法 | | public int modifyCityDistance(CityDistanceVO citydistancevo) |
| 前置条件 | | 经过验证的总经理有修改城市距离的权利，且总经理输入了有效的待修改城市距离信息 |
| 后置条件 | | 系统更新城市距离信息 |
| Manage.CityDistance.findCityDistanceBySingle | 语法 | | public ArrayList<CityDistanceVO> findCityDistanceBySingle(String city) |
| 前置条件 | | 经过验证的总经理有根据单城市查询城市距离的权利，且总经理输入了有效且存在的城市名 |
| 后置条件 | | 系统根据输入的城市名查找与其有关的城市距离，并返回城市距离信息列表 |
| Manage.CityDistance.findCityDistanceByBoth | 语法 | | public ArrayList<CityDistanceVO> findCityDistanceByBoth(String cityA, String cityB) |
| 前置条件 | | 经过验证的总经理有根据双城市查询城市距离的权利，且总经理输入了有效且存在的两个城市名 |
| 后置条件 | | 系统根据输入的两个城市名查找与其有关的城市距离，并返回城市距离信息 |
| Manage.CityDistance.showAllCityDistances | 语法 | | public ArrayList<CityDistanceVO> showAllCityDistances() |
| 前置条件 | | 经过验证的总经理有查看全部城市距离的权利 |
| 后置条件 | | 系统返回全部的城市距离信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ManageDataService.CityDistanceDataService.addCityDistance(CityDistancePO citydistancepo) | | 在管理数据包的城市距离数据文件中插入一个CityDistancePO对象 | |
| ManageDataService.CityDistanceDataService.deleteCityDistance(CityDistancePO citydistancepo) | | 在管理数据包的城市距离数据文件中删除一个CityDistancePO对象 | |
| ManageDataService.CityDistanceDataService.modifyCityDistance(CityDistancePO citydistancepo) | | 在管理数据包的城市距离数据文件中更新一个CityDistancePO对象 | |
| ManageDataService.CityDistanceDataService.findCityDistanceBySingle(String city) | | 根据输入的一个城市名，返回与它有关的CityDistancePO对象列表 | |
| ManageDataService.CityDistanceDataService.findCityDistanceByBoth(String cityA, String cityB) | | 根据输入的两个城市名，返回它们对应的CityDistancePO对象 | |
| ManageDataService.CityDistanceDataService.showAllCityDistances() | | 返回所有的CityDistancePO对象 | |

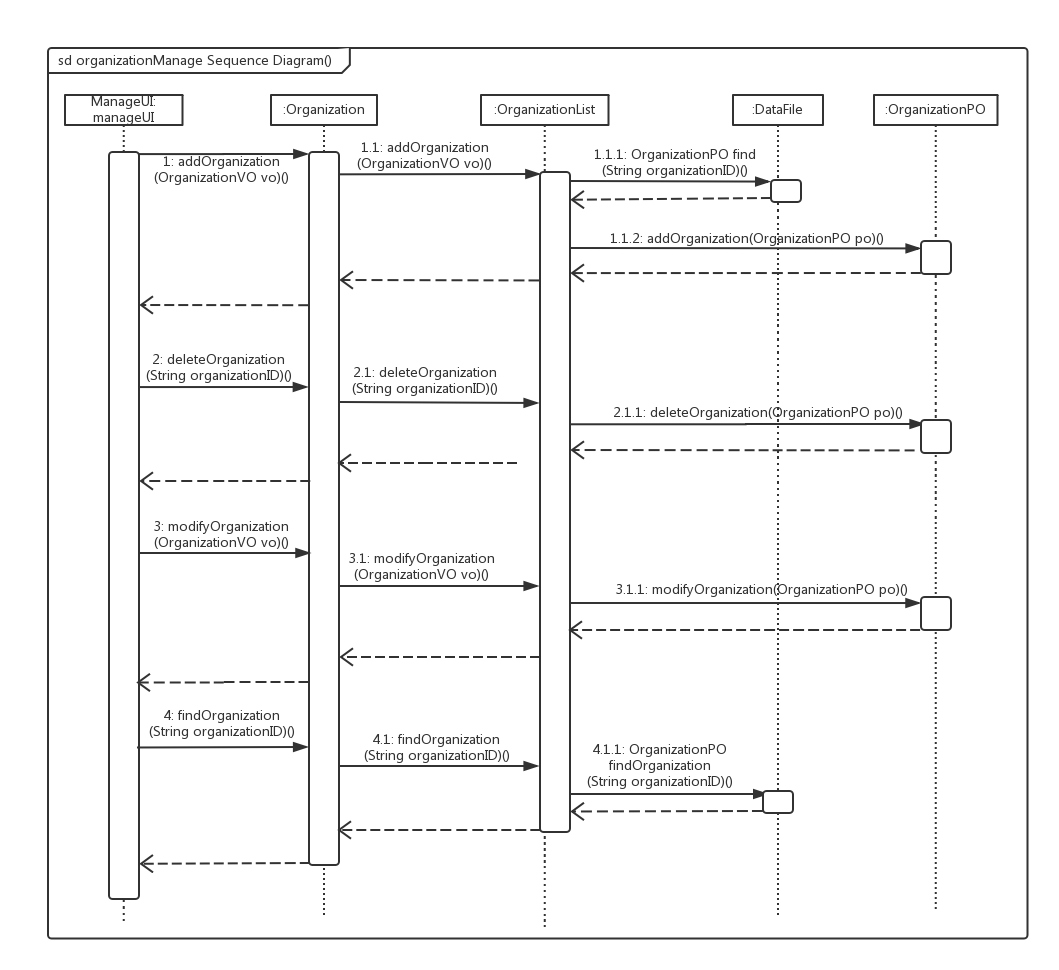
CostBL的接口规范如表7-9所示。

**表7-9 CostBL模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CostBL.modifyCost | 语法 | | public int modifyCost(CostVO costvo) |
| 前置条件 | | 经过验证的总经理有修改运费系数的权利，且总经理输入了有效的待修改运费系数信息 |
| 后置条件 | | 系统更新运费系数信息 |
| CostBL.findCost | 语法 | | public double findCost(ExpressType category) |
| 前置条件 | | 经过验证的总经理有查询运费系数的权利，且总经理输入了有效且存在的类别号 |
| 后置条件 | | 系统根据输入快递类型查找运费系数，并返回运费系数信息 |
| CostBL.showAllCosts | 语法 | | public ArrayList<CostVO> showAllCosts() |
| 前置条件 | | 经过验证的总经理有查看全部运费系数的权利 |
| 后置条件 | | 系统返回全部的运费系数信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ManageDataService.CostDataService.modifyCost(CostPO costpo) | | 在管理数据包的运费系数数据文件中更新一个CostPO对象 | |
| ManageDataService.CostDataService.findCost(ExpressType category) | | 根据输入的快递类型，返回对应的CostPO对象 | |
| ManageDataService.CostDataService.showAllCosts(CostPO costpo) | | 返回所有的CostPO对象 | |

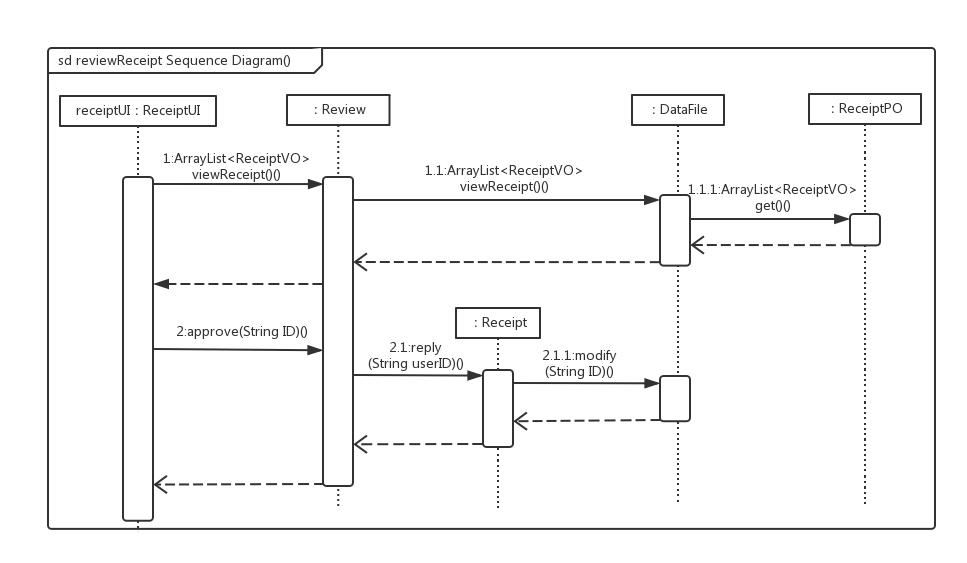
#### 4.7.2.4 业务逻辑层的动态模型

图7-2表明了快递物流系统中，当总经理开始机构管理任务后，机构管理业务逻辑处理的相关对象的协作。



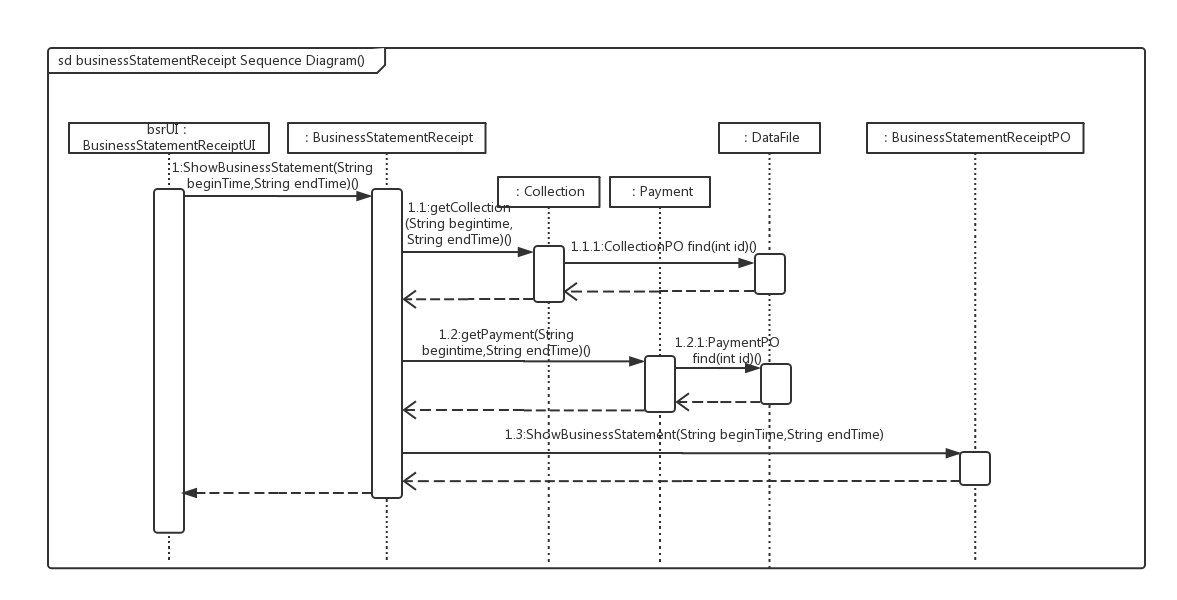
**图7-2 机构管理的顺序图**

图7-3表明了快递物流系统中，当总经理开始单据审批任务后，单据审批业务逻辑处理的相关对象的协作。



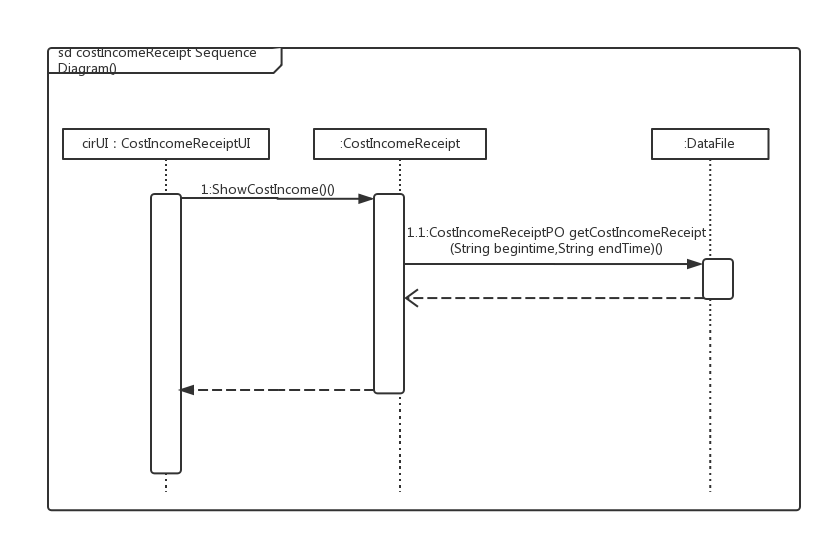
**图7-3 单据审批的顺序图**

图7-4表明了快递物流系统中，当总经理开始查看经营情况表任务后，经营情况表业务逻辑处理的相关对象的协作。



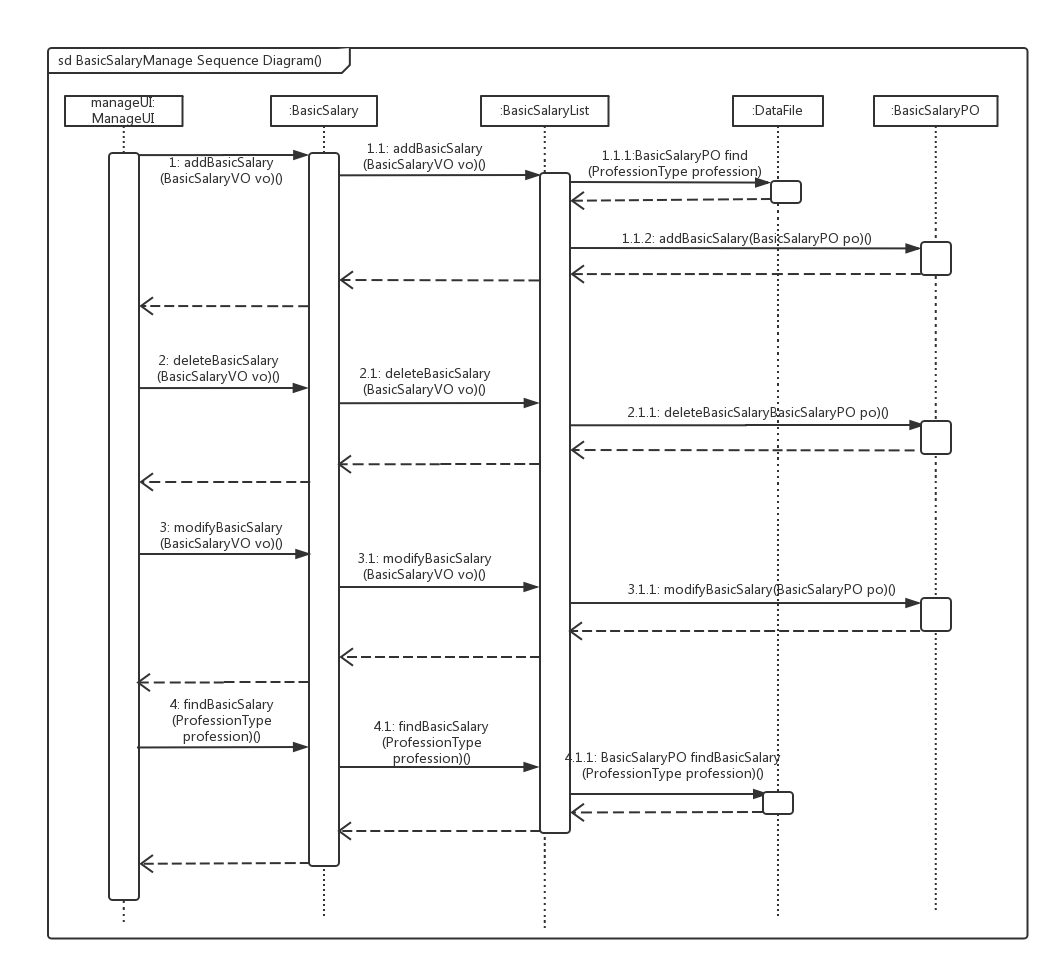
**图7-4 查看经营情况表的顺序图**

图7-5表明了快递物流系统中，当总经理开始查看成本收益表任务后，成本收益表业务逻辑处理的相关对象的协作。



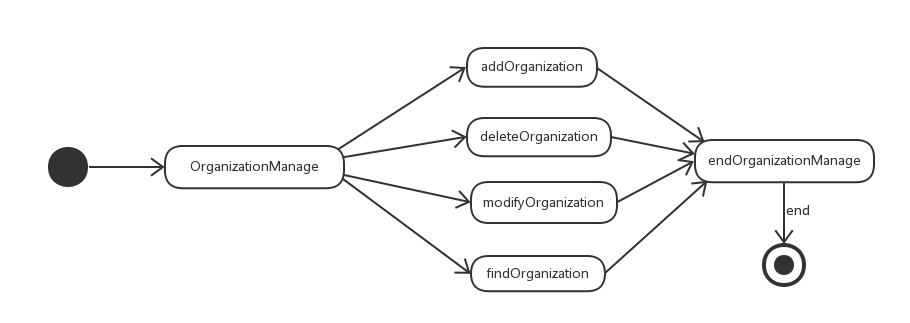
**图7-5 查看成本收益表的顺序图**

图7-6表明了快递物流系统中，当总经理开始查看基础数据设置任务后，基础数据设置业务逻辑处理的相关对象的协作（以基础月薪管理为例）。



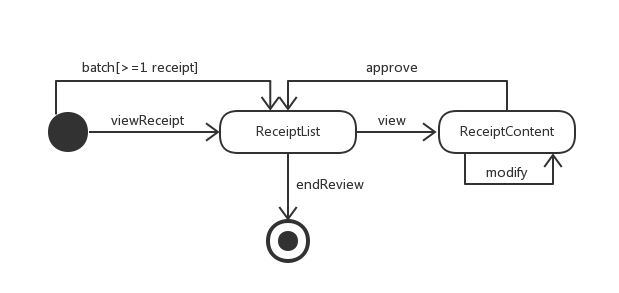
**图7-6 基础月薪管理的顺序图**

如图7-7所示的状态图描述了OrganizationBL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OrganizationManage方法被UI调用，OrganizationBL进入OrganizationManage状态；之后可以选择添加机构进入addOrganization状态，也可以选择删除机构进入deleteOrganization状态，同理modifyOrganization状态和findOrganization状态。最后结束机构管理进入endOrganizationManage状态。



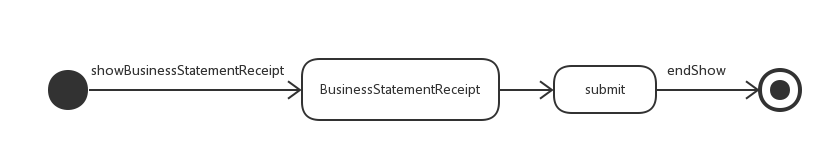
**图7-7 机构管理的状态图**

如图7-8所示的状态图描述了ReceiptBL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viewReceipt方法被UI调用，ReceiptBL进入ReceiptList状态；之后可以选择详细查看进入ReceiptContent状态，也可以选择批量审批进入ReceiptList状态。最后结束单据审批进入endReview状态。



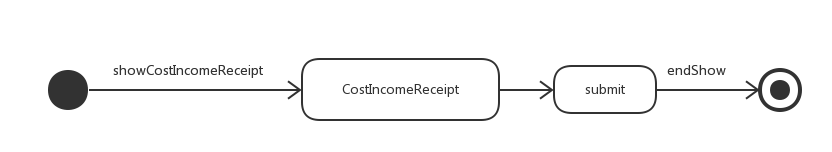
**图7-8 单据审批的状态图**

如图7-9所示的状态图描述了BusinessStatementReceiptBL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showBusinessStatementReceipt方法被UI调用，BusinessStatementReceiptBL进入BusinessStatementReceipt状态；最后通过submit方法确定查看结束，进入endShow状态。



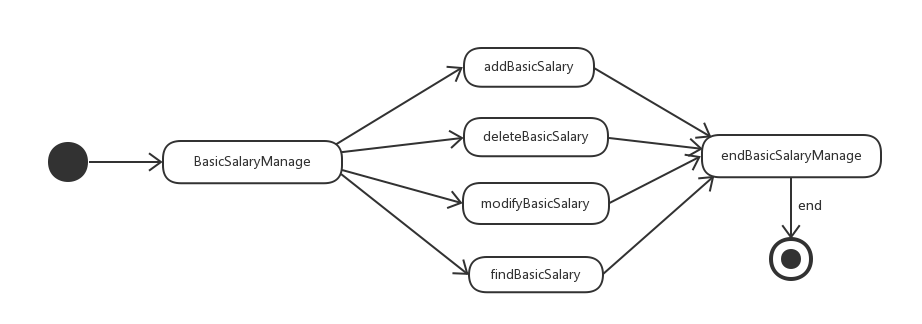
**图7-9 查看经营情况表的状态图**

如图7-10所示的状态图描述了CostIncomeReceiptBL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showCostIncomeReceipt方法被UI调用，CostIncomeReceiptBL进入CostIncomeReceipt状态；最后通过submit方法确定查看结束，进入endShow状态。



**图7-10 查看成本收益表的状态图**

如图7-7所示的状态图描述了BasicSalaryBL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BasicSalaryManage方法被UI调用，BasicSalaryBL进入BasicSalaryManage状态；之后可以选择添加基础月薪进入addBasicSalary状态，也可以选择删除基础月薪进入deleteBasicSalary状态，同理modifyBasicSalary状态和findBasicSalary状态。最后结束基础月薪管理进入endBasicSalaryManage状态。



**图7-11 基础月薪管理的状态图**

### 4.2.8userbl模块

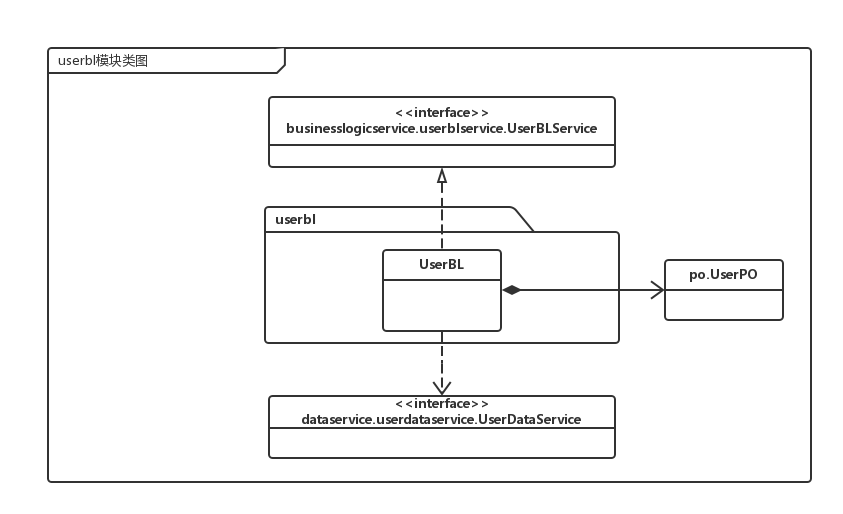
#### 4.2.8.1模块概述

userbl模块承担的需求参见ELS需求规格文档功能需求及相关非功能需求。

userBL模块的职责及接口参见ELS体系结构设计文档。

#### 4.2.8.2整体结构

根据体系结构的设计，我们将系统分为展示层，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。比如展示层和业务逻辑层之间，我们添加userblservice.UserBLService接口。业务逻辑层和数据层之间添加uesrdataservice.UserDataService接口。ReceiptPO是作为所有单据父类的持久化对象被添加到设计模型中去的。ReceiptBL模块的设计如图8-1所示。



**图8-1 userbl模块各个类的设计**

userbl模块各个类的职责如表8-1所示。

**表8-1 userbl模块个各类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserBL | 负责所有用户登录以及管理员增删改查用户的操作 |

#### 4.2.8.3模块内部类的接口规范

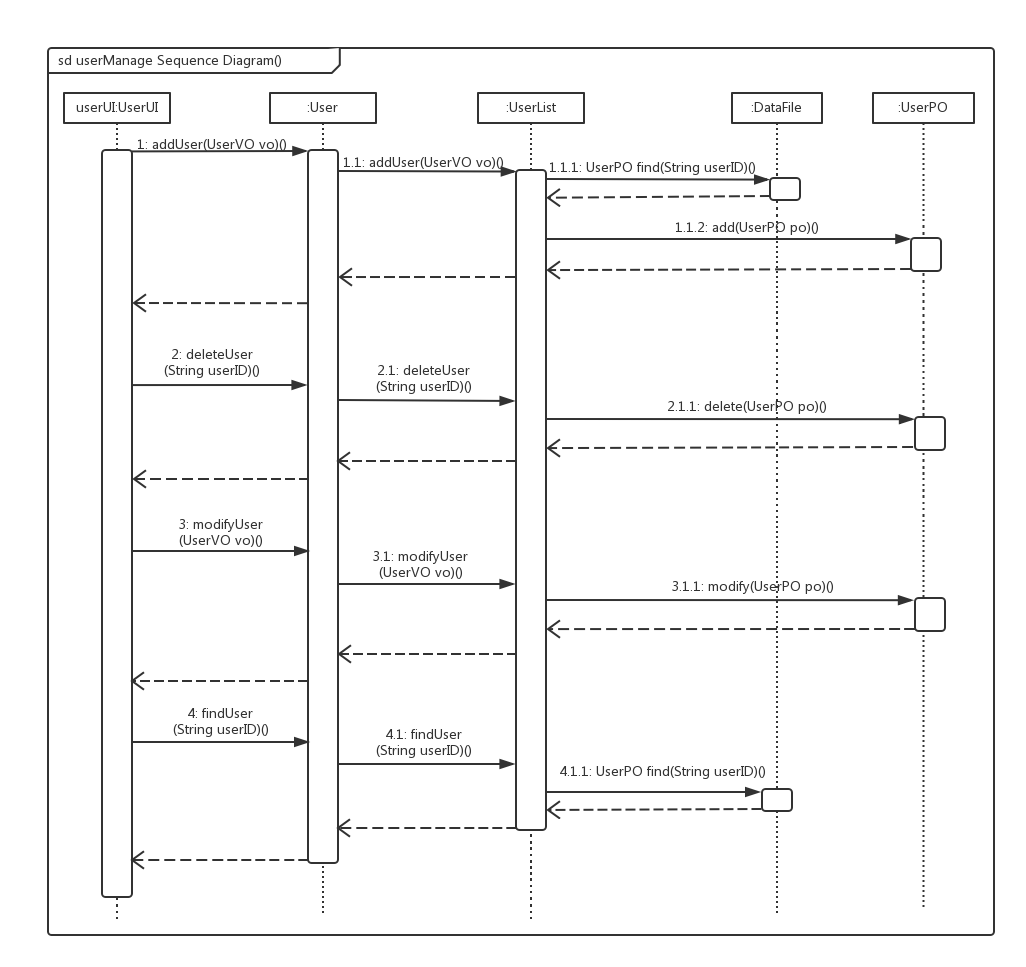
UserBL的接口规范如表8-2所示。

**表8-2 UserBL的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.login | 语法 | public int login(String userID, String password) |
| 前置条件 | 快递物流系统的用户在登录界面输入了自己的用户编号和密码 |
| 后置条件 | 查找是否存在编号为userID的用户，根据输入的password返回登录验证的结果 |
| User.addUser | 语法 | public int addUser(UserVO uservo) |
| 前置条件 | 经过验证的管理员有添加新用户的权利，且管理员输入了完整有效的新用户信息 |
| 后置条件 | 系统保存新增的用户信息并更新用户列表 |
| User.deleteUser | 语法 | public int deleteUser(String userID) |
| 前置条件 | 经过验证的管理员有删除用户的权利，且管理员输入了有效的待删除用户信息 |
| 后置条件 | 系统查找对应的用户信息，并删除该用户对象 |
| User.modifyUserPassword | 语法 | public int modifyUserPassword (String userID, String newPassword) |
| 前置条件 | 系统中每个用户有修改自己密码的权利，且该用户输入了有效的新密码 |
| 后置条件 | 系统根据用户编号和输入的新密码更新用户信息 |
| User.modifyUserAuthority | 语法 | public int modifyUserAuthority(String userID, AuthorityType authority) |
| 前置条件 | 经过验证的管理员有更改用户权限的权利，且管理员输入了有效的待更改用户信息 |
| 后置条件 | 系统根据用户编号和新的用户权限更新用户信息 |
| User.findUser | 语法 | public UserVO findUser(String userID) |
| 前置条件 | 经过验证的管理员或总经理有查询用户的权利，且管理员或总经理输入了有效的待查询用户编号 |
| 后置条件 | 系统根据输入的用户编号查找用户，并返回用户信息 |
| User.findUserByKeyword | 语法 | public ArrayList<UserVO> findUserByKeyword(String keyword) |
| 前置条件 | 经过验证的管理员或总经理有按关键字查询用户的权利，且管理员或总经理输入了有效的关键字 |
| 后置条件 | 系统根据输入的关键字查找用户，并返回与之相关的用户信息列表 |
| User.showAllUsers | 语法 | public ArrayList<UserVO> showAllUsers() |
| 前置条件 | 经过验证的管理员或总经理有查看全部用户的权利 |
| 后置条件 | 系统返回全部的用户信息 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| FileFactory.getUserData | | 得到User文件的引用 |
| UserDataService.check(String userID ,String password) | | 得到根据输入的ID和password验证的结果 |
| UserDataService.addUser(UserPO userpo) | | 在用户数据文件中插入一个UserPO对象 |
| UserDataService.deleteUser(String userID) | | 在用户数据文件中删除一个UserPO对象 |
| userDataService.modify(UserPO userpo) | | 在用户数据文件中更新一个UserPO对象 |
| userDataService.findUserByID (String userID) | | 根据输入的用户编号，返回对应的UserPO对象 |
| userDataService.findUserByKeyword (String keyword) | | 根据输入的关键字，返回相关的UserPO对象列表 |
| userDataService.showAll () | | 返回所有的UserPO对象 |

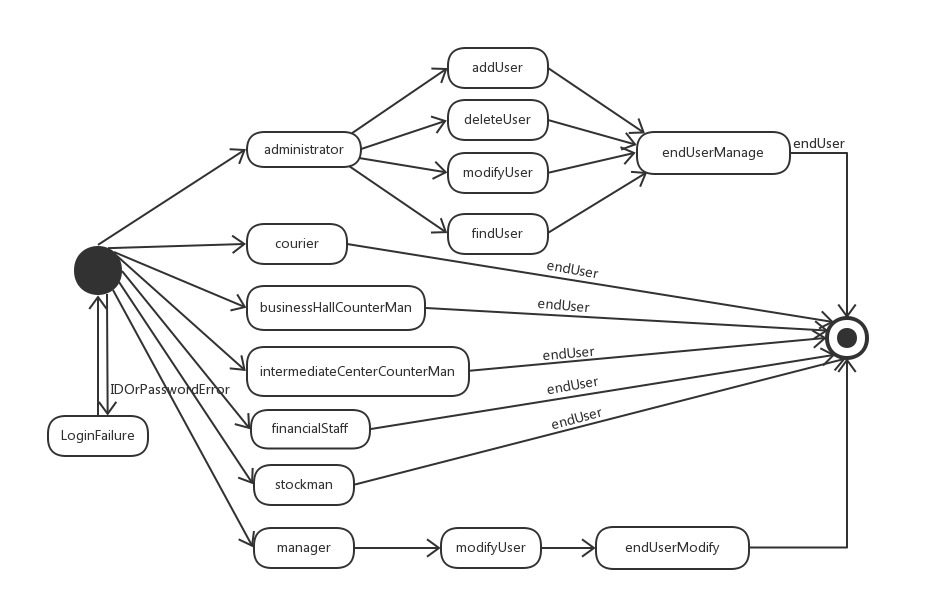
#### 4.2.8.4业务逻辑层的动态模型

图8-2表明了ELS快递物流系统中，当用户选择登录或管理员选择用户增删改查的基本操作之后，用户业务逻辑处理的相关对象之间的协作。



**图8-2 用户登录及管理员增删改查的顺序图**

如图8-3所示，用户登录及管理员增删改查对象状态图描述了User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establishUser方法被 UI调用，establishUser根据不同的指令进入不同状态。



**图8-3 用户登录及管理员增删改查的状态图**

## 4.3数据层的分解

### 4.3.1模块职责

数据层模块的职责如表3所示。

**表3 数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ExpressDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| BusinessDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| IntermediateDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| FinanceDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| RepertoryDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| ManageDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| UserDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| ReceiptDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |

### 4.3.2接口规范

express数据层模块的接口规范如表4所示。

**表4 express数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExpressDataService.getBaseCost | 语法 | public CostBasePO getBaseCost() |
| 前置条件 | 系统中存在订单收费的价格信息信息 |
| 后置条件 | 系统返回订单收费的价格信息 |
| ExpressDataService.chargeCollection | 语法 | public boolean chargeCollection(ExpressPO po) |
| 前置条件 | ExpressPO中存在成员charge需要更改 |
| 后置条件 | 系统更新ExpressPO的charge属性 |
| ExpressDataService.getExpressInfos | 语法 | public ExpressPO getExpressInfos() |
| 前置条件 | 系统中存在ExpressPO的信息 |
| 后置条件 | 系统返回ExpressPO |

business数据层模块的接口规范如表5所示。

**表5 business数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BusinessDataService.addVehicle | 语法 | public boolean addVehicle(VehiclePO po) |
| 前置条件 | VehiclePOList中需要添加一个VehiclePO |
| 后置条件 | 系统在VehiclePOList中添加一个VehiclePO并返回操作结果 |
| BusinessDataService.getFreeVehicle | 语法 | public ArrayList<VehiclePO> getVehicleInfos() |
| 前置条件 | 系统存在空闲的车辆信息VehiclePO |
| 后置条件 | 系统返回空闲的车辆信息列表VehiclePOList |
| BusinessDataService.deleteVehicle | 语法 | public boolean BusinessDataService.deleteVehicle(VehiclePO po) |
| 前置条件 | 系统中存在改车辆信息VehiclePO |
| 后置条件 | 系统删除该车辆信息，并返回操作结果 |
| BusinessDataService.modifyVehicle | 语法 | public boolean BusinessDataService.modifyVehicle(VehiclePO po) |
| 前置条件 | 系统中存在该车辆信息VehiclePO |
| 后置条件 | 系统修改该车辆信息，并返回操作结果 |
| BusinessDataService.getDriverInfo | 语法 | public DriverPO getDriverInfos(String ID) |
| 前置条件 | 系统中存在该司机信息DriverPO |
| 后置条件 | 系统返回该司机信息DriverPO |
| BusinessDataService.addDriver | 语法 | public boolean addDriver(DriverPO po) |
| 前置条件 | DriverPOList中需要添加一个DriverPO |
| 后置条件 | 系统在DriverPOList中添加一个DriverPO并返回操作结果 |
| BusinessDataService.deleteDriver | 语法 | public boolean deleteDriver(DriverPO po) |
| 前置条件 | 系统中存在该司机信息DriverPO |
| 后置条件 | 系统删除该司机信息DriverPO，并返回操作结果 |
| BusinessDataService.modifyDriver | 语法 | public boolean modifyDriver(DriverPO po) |
| 前置条件 | 系统中存在该车辆信息VehiclePO |
| 后置条件 | 系统修改该车辆信息，并返回操作结果 |
| BusinessDataService.getTransferOrders | 语法 | public ArrayList<OrderVO>getTransferOrders() |
| 前置条件 | 系统中存在待转运的的OrderVO |
| 后置条件 | 系统返回待转运的的OrderVO的列表OrderVOList |
| BusinessDataService.getSendOrder | 语法 | public ArrayList<VehiclePO> getFreeVehicles() |
| 前置条件 | 系统中存在待派送的的OrderVO |
| 后置条件 | 系统返回待派送的的OrderVO的列表OrderVOList |

intermediate数据层模块的接口规范如表6所示。

**表6 intermediate数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| IntermediateDataService. showTransferingReceipt | 语法 | Public TransferingReceiptPO getTransferingReceipt(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TransferingReceiptPO对象 |
| IntermediateDataService.addOrder | 语法 | public boolean addOrder(String ID); |
| 前置条件 | 同样的ID在TransferingReceiptPO的对象中不存在 |
| 后置条件 | 在外部文件中增加一个OrderPO对象 |
| IntermediateDataService.deleteOrder | 语法 | public boolean deleteOrder(String ID); |
| 前置条件 | 在外部文件中存在一个ID符合的OrderPO对象需要删除 |
| 后置条件 | 删除一个OrderPO对象 |
| IntermediateDataService.saveTransferingReceipt | 语法 | public void updateTransferingReceipt(TransferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 在外部文件中增加一个TransferingReceiptPO对象 |
| IntermediateDataService.getPlaneList | 语法 | public ArrayList<PlanePO> getPlaneList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储PlanePO的ArrayList对象 |
| IntermediateDataService. getPlane | 语法 | public PlanePO getPlane(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个PlanePO对象 |
| IntermediateDataService.updateWaitingList | 语法 | public ArrayList<OrderPO> updateWaitingList(TranferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 存储OrderPO对象的ArrayList对象更新 |
| IntermediateDataService. Envehicle | 语法 | public EnplaningReceiptPO enplane(ArrayList<OrderPO>); |
| 前置条件 | 存储OrderPO对象的ArrayList对象更新，且不为空链表 |
| 后置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| IntermediateDataService. saveEnplaningReceiptList | 语法 | public ArrayList<EnplaningReceiptPO> updateEnplaningReceiptList(EnplaningReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EnplaningReceiptPO对象 |
| IntermediateDataService. saveFare | 语法 | public boolean updateFare(farePO); |
| 前置条件 | 同样的farePO对象在外部文件中不存在 |
| 后置条件 | 外部文件中新增一个farePO对象 |
| IntermediateDataService.getEnplaningReceipt | 语法 | Public boolean getEnplaningReceipt(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | 存储EnplaningReceiptPO对象的ArrayList对象不为空链表，且同样的EnplaningReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. saveEnplaningReceipt | 语法 | Public boolean updateEnplaningReceipt(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EnplaningRecePO对象 |
| IntermediateDataService.getTrainList | 语法 | public ArrayList<TrainPO> getTrainList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储TrainPO的ArrayList对象 |
| IntermediateDataService. getTrain | 语法 | public TrainPO getTrain(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TrainPO对象 |
| IntermediateDataService. saveEntrainingReceiptList | 语法 | public ArrayList<EntrainingReceiptPO> updateEntrainingReceiptList(EntrainingReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EntrainingReceiptPO对象 |
| IntermediateDataService.getEntrainingReceipt | 语法 | public boolean getEntrainingReceipt(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | 存储EntrainingReceiptPO对象的ArrayList对象不为空链表，且同样的EntrainingReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. saveEntrainingReceipt | 语法 | public boolean updateEntrainingReceipt(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EntrainingRecePO对象 |
| IntermediateDataService.getTruckList | 语法 | public ArrayList<TruckPO> getTruckList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储TruckPO的ArrayList对象 |
| IntermediateDataService. getTruck | 语法 | public TruckPO getTruck(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TruckPO对象 |
| IntermediateDataService. updateEntruckingReceiptList | 语法 | public ArrayList<EntruckingReceiptPO> updateEntruckingReceiptList(EntruckingReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EntruckingReceiptPO对象 |
| IntermediateDataService.getEntruckingReceipt | 语法 | public boolean getEntruckingReceipt(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | 存储EntruckingReceiptPO对象的ArrayList对象不为空链表，且同样的EntruckingReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. updatEntruckingReceipt | 语法 | public boolean updateEntruckingReceipt(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EntruckingRecePO对象 |

finance数据层模块的接口规范如表7所示。

**表7 finance数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceDataService.InitInfo | 语法 | public int InitInfo(InitInfoPO po)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个 InitInfoPO |
| FinanceDataService.getInitInfo | 语法 | public InitInfoPO getInitInfo（String time）throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回一个 InitInfoPO |
| FinanceDataService.addAccount | 语法 | public int addAccount(AccountPO po) throws RemoteException; |
| 前置条件 | 同样ID的AccountPO在数据库中不存在 |
| 后置条件 | 在数据库中增加一个AccountPO |
| FinanceDataService.deleteAccount | 语法 | public int deleteAccount(AccountPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的AccountPO |
| 后置条件 | 删除一个AccountPO |
| FinanceDataService.modifyAccount | 语法 | public int modifyAccount(AccountPO po,String name) throws RemoteException; |
| 前置条件 | 在数据库中存在某个需要被修改的AccountPO |
| 后置条件 | 修改一个AccountPO |
| FinanceDataService.findByName | 语法 | public AccountPO findbyName(String name) throws RemoteException； |
| 前置条件 | 含有该名称的AccountPO在数据库中存在 |
| 后置条件 | 按名称进行查找返回相应的AccountPO |
| FinanceDataService.findByKeyword | 语法 | public ArrayList<AccountPO> findByKeyword(String s) throws RemoteException； |
| 前置条件 | 含有关键字的AccountPO在数据库中存在 |
| 后置条件 | 按关键字进行查找返回相应的AcountPO的ArrayList |
| FinanceDataService.showAll | 语法 | Public ArrayList<AccountPO> showAll() throws RemoteException; |
| 前置条件 | 在数据库中存在AccountPO的记录 |
| 后置条件 | 返回数据库中所有的AccountPO记录 |
| FinanceDataService.showBSL | 语法 | public BusinessstatementReceiptPO showBSL(String beginTime,String endTime) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 BusinessstatementReceiptPO |
| 后置条件 | 返回一个BusinessstatementReceiptPO |
| FinanceDataService.showAllBSList | 语法 | public ArrayList<BusinessstatementReceiptPO> showAllBSList()throws RemoteException; |
| 前置条件 | 数据库中存在 BusinessstatementReceiptPO |
| 后置条件 | 返回 BusinessstatementReceiptPO的ArrayList |
| FinanceDataService.export | 语法 | public int export(BusinessstatementReceiptPO po)throws RemoteException; |
| 前置条件 | 数据库中存在 BusinessstatementReceiptPO |
| 后置条件 | 导出 BusinessstatementReceiptPO到excel |
| FinanceDataService.creatCollection | 语法 | public int createCollection(CollectionReceiptPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张入款单 CollectionReceiptPO |
| FinanceDataService.getAllCollection | 语法 | public ArrayList<CollectionReceiptPO> getAllCollection() throws RemoteException; |
| 前置条件 | 数据库中存在 CollectionReceiptPO记录 |
| 后置条件 | 返回数据库中所有的 CollectionReceiptPO记录 |
| FinanceDataService.getGathering | 语法 | public ArrayList<GatheringReceiptPO> getGathering(String HallID,String Time) throws RemoteException; |
| 前置条件 | 数据库中存在 GatheringReceiptPO记录 |
| 后置条件 | 返回符合条件的 GatheringReceiptPO记录 |
| FinanceDataService.getMoney | 语法 | public double[] getMoney(GatheringReceiptPO po) throws RemoteException; |
| 前置条件 | 已经获得GatheringReceiptPO记录 |
| 后置条件 | 返回GatheringReceiptPO中所有金额 |
| FinanceDataService.getTotalMoney | 语法 | public double getTotalMoney(int[] money) throws RemoteException; |
| 前置条件 | 已经获得GatheringReceiptPO中的所有金额 |
| 后置条件 | 返回所有金额之和 |
| FinanceDataService.getNum | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已创建的入款单数 |
| FinanceDataService.findByID | 语法 | public CollectionReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的CollectionReceiptPO |
| 后置条件 | 返回相应ID的 CollectionReceiptPO记录 |
| FinanceDataService.modify | 语法 | publicCollectionReceiptPOmodify(CollectionReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 CollectionReceiptPO |
| 后置条件 | 修改相应的 CollectionReceiptPO记录 |
| FinanceDataService.creatPaymentReceipt(PaymentReceiptPO po) | 语法 | public int creatPaymentReceipt(PaymentReceiptPO vo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张付款单PaymentReceiptPO |
| FinanceDataService.getAllPaymentReceipt() | 语法 | public ArrayList<PaymentReceiptPO> getAllPaymentReceipt() throws RemoteException; |
| 前置条件 | 数据库中存在 PaymentReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 PaymentReceiptPO记录 |
| FinanceDataService. getNum() | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已创建的付款单数 |
| FinanceDataService.findByID(String ID) | 语法 | public PaymentReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合ID的 PaymentReceiptPO |
| 后置条件 | 返回相应的 PaymentReceiptPO记录 |
| FinanceDataService.modify(PaymentReceiptPO po) | 语法 | public PaymentReceiptPO modify(PaymentReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的PaymentReceiptPO |
| 后置条件 | 修改相应的PaymentReceiptPO记录 |
| FinanceDataService.creatCostIncomeList(CostIncomeReceiptPO po) | 语法 | public int creatCostIncomeList(CostIncomeReceiptPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张成本收益表CostIncomeReceiptPO |
| FinanceDataService.getAllCostIncomeList() | 语法 | public ArrayList<CostIncomeReceiptPO> getAllCostIncomeList() throws RemoteException; |
| 前置条件 | 数据库中存在 CostIncomeReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 CostIncomeReceiptPO记录 |
| FinanceDataService.getCollection() | 语法 | public ArrayList<CollectionReceiptPO> getCollection() throws RemoteException; |
| 前置条件 | 数据库中存在 CollectionReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 CollectionReceiptPO记录 |
| FinanceDataService.getPayment() | 语法 | public ArrayList<PaymentReceiptPO>getPayment() throws RemoteException; |
| 前置条件 | 数据库中存在 PaymentReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 PaymentReceiptPO记录 |
| FinanceDataService. getNum() | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已经创建的成本收益表数 |
| FinanceDataService.findByID(String ID) | 语法 | public CostIncomeReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合ID的CostIncomeReceiptPO记录 |
| 后置条件 | 返回相应的 CostIncomeReceiptPO记录 |
| FinanceDataService.modify(CostIncomeReceiptPO po) | 语法 | public CostIncomeReceiptPO modify(CostIncomeReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 CostIncomeReceiptPO |
| 后置条件 | 修改相应的CostIncomeReceiptPO记录 |

repertory数据层模块的接口规范如表8所示。

**表8 repertory数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RepertoryDataService. modifyRepertory | 语法 | public int modifyRepertory(RepertoryPO repertorypo) |
| 前置条件 | 需要修改仓库信息 |
| 后置条件 | 更新一个po对象 |
| RepertoryDataService. findRepertory | 语法 | public RepertoryPO findRepertory(String repertoryID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的RepertoryPO的结果 |
| RepertoryDataService. addGoods | 语法 | public int addGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 无 |
| 后置条件 | 在仓库的库存信息列表中增加一个po记录 |
| RepertoryDataService. deleteGoods | 语法 | public int deleteGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 在仓库的库存信息列表中存在某一个po需要删除 |
| 后置条件 | 删除一个po对象 |
| RepertoryDataService. modifyGoods | 语法 | public int modifyGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 在仓库的库存信息列表中存在某一个po需要更新 |
| 后置条件 | 更新一个po对象 |
| RepertoryDataService. findGoodsbyID | 语法 | public GoodsPO findGoodsbyID(String repertoryID, String JJD-ID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的GoodsPO的结果 |
| RepertoryDataService.findGoodsbyDate | 语法 | public GoodsPO findGoodsbyDate(String repertoryID, String beginDate, String endDate) |
| 前置条件 | 无 |
| 后置条件 | 按beginDate和endDate进行查找返回相应的GoodsPO的结果 |
| RepertoryDataService.findGoodsbyTime | 语法 | public GoodsPO findGoodsbyTime(String repertoryID, String time) |
| 前置条件 | 无 |
| 后置条件 | 按time进行查找返回相应的GoodsPO的结果 |

manage数据层模块的接口规范如表9所示。

**表9manage数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageDataService.addOrganization | 语法 | public int addOrganization(OrganizationPO organizationpo) |
| 前置条件 | 同样ID的organizationpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个organizationpo记录 |
| ManageDataService.deleteOrganization | 语法 | public int deleteOrganization(OrganizationPO organizationpo) |
| 前置条件 | 在外部文件中存在某一个organizationpo需要删除 |
| 后置条件 | 删除一个organizationpo对象 |
| ManageDataService.modifyOrganization | 语法 | public int modifyOrganization(OrganizationPO organizationpo) |
| 前置条件 | 在外部文件中存在某一个organizationpo需要更新 |
| 后置条件 | 更新一个organizationpo对象 |
| ManageDataService.findOrganization | 语法 | public OrganizationPO findOrganization(String organizationID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的OrganizationPO的结果 |
| ManageDataService.modifyUser | 语法 | public int modifyUser(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个userpo需要更新 |
| 后置条件 | 更新一个userpo对象 |
| ManageDataService.addPerWage | 语法 | public int addPerWage(PerWagePO perwagepo) |
| 前置条件 | 同样profession的perwagepo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个perwagepo记录 |
| ManageDataService.deletePerWage | 语法 | public int deletePerWage(PerWagePO perwagepo) |
| 前置条件 | 在外部文件中存在某一个perwagepo需要删除 |
| 后置条件 | 删除一个perwagepo对象 |
| ManageDataService.modifyPerWage | 语法 | public int modifyPerWage(PerWagePO perwagepo) |
| 前置条件 | 在外部文件中存在某一个perwagepo需要更新 |
| 后置条件 | 更新一个perwagepo对象 |
| ManageDataService.findPerWage | 语法 | public double findPerWage(int profession) |
| 前置条件 | 无 |
| 后置条件 | 按profession进行查找返回相应的PerWage的值 |
| ManageDataService.addBasicSalary | 语法 | public int addBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 同样profession的basicsalarypo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个basicsalarypo记录 |
| ManageDataService.deleteBasicSalary | 语法 | public int deleteBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 在外部文件中存在某一个basicsalarypo需要删除 |
| 后置条件 | 删除一个basicsalarypo对象 |
| ManageDataService.modifyBasicSalary | 语法 | public int modifyBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 在外部文件中存在某一个basicsalarypo需要更新 |
| 后置条件 | 更新一个basicsalarypo对象 |
| ManageDataService.findBasicSalary | 语法 | public double findBasicSalary((int profession)) |
| 前置条件 | 无 |
| 后置条件 | 按profession进行查找返回相应的BasicSalary的值 |
| ManageDataService.addCityDistance | 语法 | public int addCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 同样City A, City B的citydistancepo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个citydistancepo记录 |
| ManageDataService.deleteCityDistance | 语法 | public int deleteCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 在外部文件中存在某一个citydistancepo需要删除 |
| 后置条件 | 删除一个citydistancepo对象 |
| ManageDataService.modifyCityDistance | 语法 | public int modifyCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 在外部文件中存在某一个citydistancepo需要更新 |
| 后置条件 | 更新一个citydistancepo对象 |
| ManageDataService.findCityDistance | 语法 | public double findCityDistance(String CityA, String CityB) |
| 前置条件 | 无 |
| 后置条件 | 按CityA, CityB进行查找返回相应的CityDistance的值 |
| ManageDataService.addCost | 语法 | public int addCost(CostPO costpo) |
| 前置条件 | 同样category的costpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个costpo记录 |
| ManageDataService.deleteCost | 语法 | public int deleteCost(CostPO costpo) |
| 前置条件 | 在外部文件中存在某一个costpo需要删除 |
| 后置条件 | 删除一个costpo对象 |
| ManageDataService.modifyCost | 语法 | public int modifyCost(CostPO costpo) |
| 前置条件 | 在外部文件中存在某一个costpo需要更新 |
| 后置条件 | 更新一个costpo对象 |
| ManageDataService.findCost | 语法 | public double findCost(category) |
| 前置条件 | 无 |
| 后置条件 | 按category进行查找返回相应的Cost的值 |

user数据层模块的接口规范如表10所示。

**表10 user数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService.check | 语法 | public boolean check(String userID, String password) |
| 前置条件 | 无 |
| 后置条件 | 返回用户名和密码是否匹配 |
| UserDataService.add | 语法 | public int add(UserPO userpo) |
| 前置条件 | 同样ID的userpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个po记录 |
| UserDataService.delete | 语法 | public int delete(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个po需要删除 |
| 后置条件 | 删除一个userpo对象 |
| UserDataService.modify | 语法 | public int modify(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个po需要更新 |
| 后置条件 | 更新一个userpo对象 |
| UserDataService.find | 语法 | public UserPO find (String ID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的UserPO的结果 |
| UserDataService.showAll | 语法 | public ArrayList<UserPO> showAll() |
| 前置条件 | 无 |
| 后置条件 | 返回所有的UserPO的结果 |

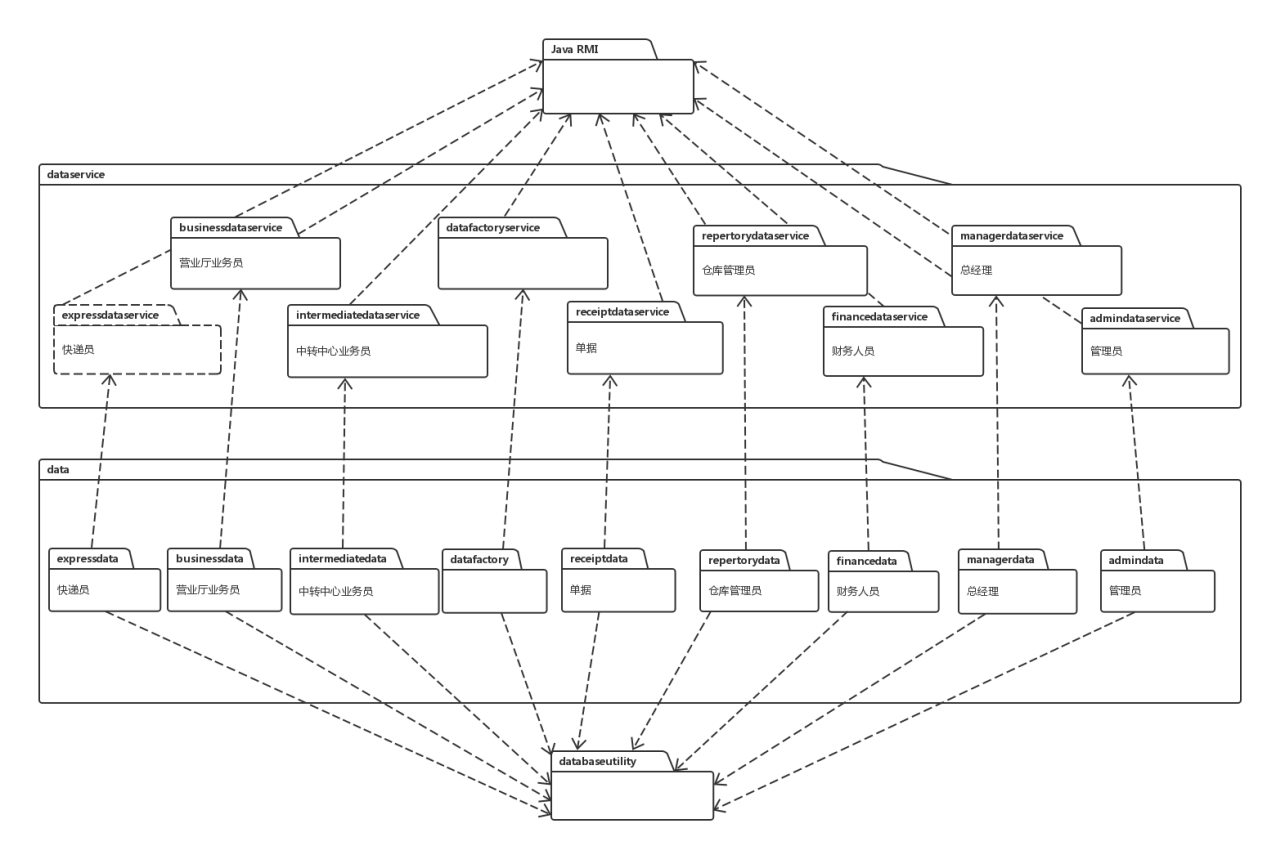
receipt数据层模块的接口规范如表11所示。

**表11 receipt数据层模块的接口规范**

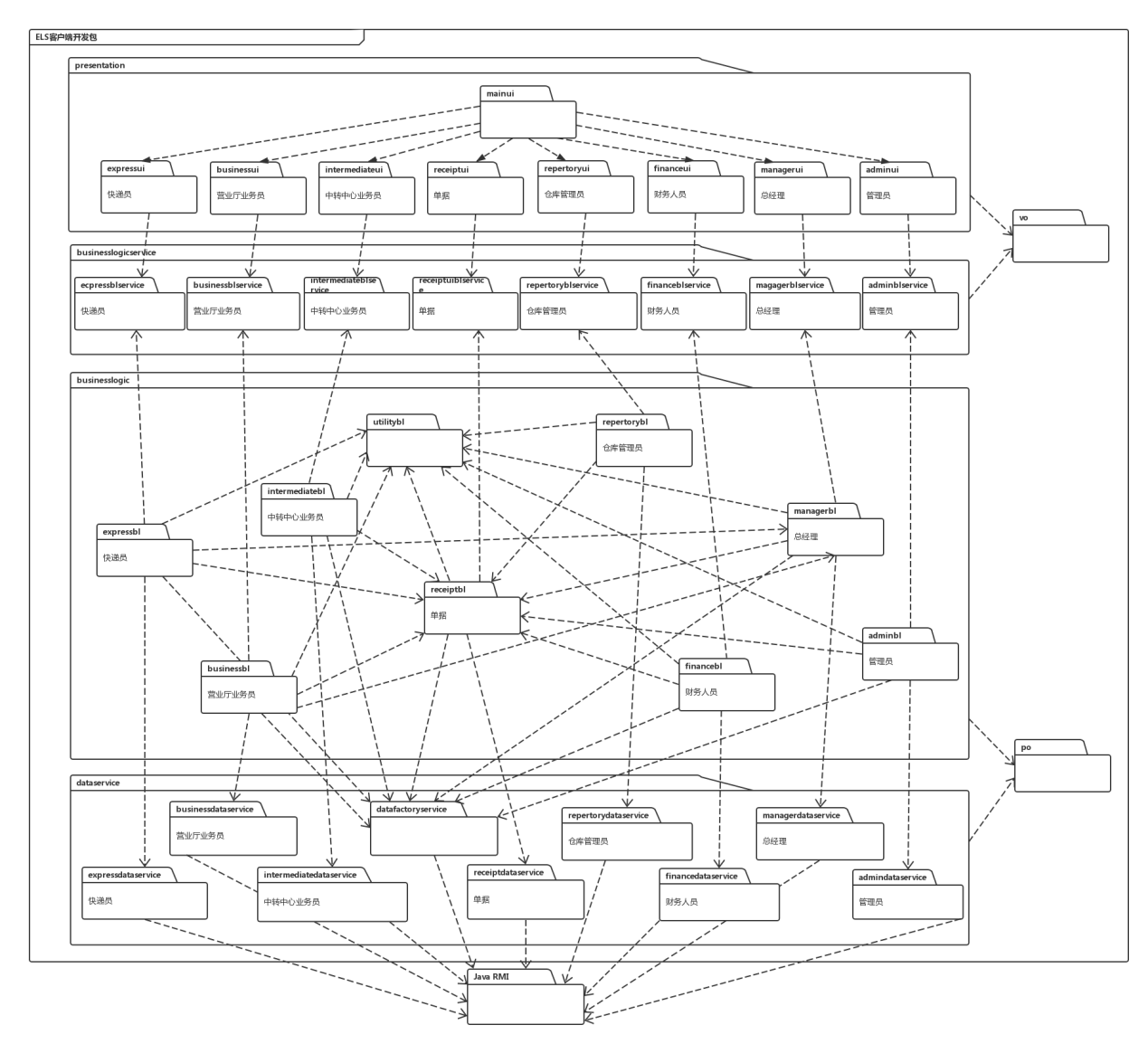
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| ReceiptDataService.add | 语法 | Public int Add(ReceiptPo po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个po记录 |
| ReceiptDataService.find | 语法 | Public ReceiptPO find(String id)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的ReceiptPo()结果 |
| ReceiptDataService.modify | 语法 | Public int Modify(ReceiptPo po)throws RemoteException |
| 前置条件 | 该单据id存在 |
| 后置条件 | 修改一个po |
| ReceiptDataService.show | 语法 | Public ArrayList<ReceiptPO> show()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有的po |
| ReceiptDataService.init | 语法 | Public void init()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始持久化数据库的使用 |

# 5.依赖视角

图1,2是服务器端和客户端各自的包之间的依赖关系。



**图1 ELS服务器端开发包图**



**图2 ELS客户器端开发包图**