

快递管理系统软件详细设计文档



2015-11-7

南京大学软件学院

林庆、张健、李芷牧、令佩棠

目录

[1. 引言 3](#_Toc434702519)

[1.1编制目的 3](#_Toc434702520)

[1.2词汇表 3](#_Toc434702521)

[1.3参考资料 3](#_Toc434702522)

[2. 产品概述 3](#_Toc434702523)

[3. 体系结构设计概述 4](#_Toc434702524)

[4. 结构视角 4](#_Toc434702525)

[4.1 业务逻辑层的分解 4](#_Toc434702526)

[4.1.1 storagebl模块 4](#_Toc434702527)

[4.1.2 logispicsquerybl模块 11](#_Toc434702528)

[4.1.3 collectionbl模块 11](#_Toc434702529)

[4.1.4 transferbl模块 13](#_Toc434702530)

[4.1.5 approvelbl模块 17](#_Toc434702531)

[4.1.6 financebl模块 17](#_Toc434702532)

[4.1.7 infobl模块 17](#_Toc434702533)

[4.1.8 systembl模块 25](#_Toc434702534)

[5. 依赖视角 28](#_Toc434702535)

# 1. 引言

## 1.1编制目的

本报告详细完成了对快递物流管理系统的详细设计，达到指导后续软件构造的目的，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 |
| YloadDoc | 营业厅装车单 |  |
| YArrivalDoc | 营业厅到达单 |  |
| ZArrivalDoc | 中转中心到达单 |  |
| TransferDoc | 中转单 |  |
| ZloadDoc | 中转中心装车单 |  |
| SendDoc | 寄件单 |  |
| InWarehouseDoc | 入库单 |  |
| OutWarehouseDoc | 出库单 |  |
| OverDoc | 收件单 |  |
| PayDoc | 付款单 |  |
| GatheringDoc | 收款单 |  |
| YDeliverDoc | 派件单 |  |
| CostInComeDocPO | 成本收益表 |  |
| RecordPO | 仓库记录 |  |
| CMASSS | 快递管理与服务系统 |  |

## 1.3参考资料

1) 快递管理与服务系统（CMASS）用例文档v1.0

2) 快递管理与服务系统（CMASS）软件需求规格说明文档

# 2. 产品概述

CMASS是为一家本地民营物流企业开发的快递管理与服务系统，开发的目标是帮助企业wanchengrichang 的业务流程，包括：

SF1：对输送、保管、包装、流通加工等快递物流活动进行衔接活动，对装卸活动的管理，确定最恰当的装卸方式，合理配置。获得较好的经济效果。

SF2：对运输活动的管理，要求选择技术经济效果最好的运输方式及联运方式，合理确定运输路线，以实现安全、迅速、准时、价廉的要求。

SF3：处理营业厅司机，车辆的管理。

SF4：对快件的运费和到达时间有预估功能。

SF5：处理中转中心或营业厅快递的接收与派件，包括装车管理，装运管理等。

SF6：对仓库的管理，包括仓库货物的出库，入库，报警，分区，盘点等。

SF7：帮助财务人员处理公司财务状况，包括账户管理，制作成本收益，经营情况表等。

SF8：协助总经理的决策。

通过CMASS的应用，期望为民营企业提高员工工作效率，降低成本，减少企业内部交流障碍，吸引顾客，提高利润。

# 3. 体系结构设计概述

对于本快递物流信息管理系统（CMASS），系统将由两部分程序组成，一部分是安装在不同机构的客户端程序，另一部分是服务器端程序。采用分层风格完成对功能性需求的设计，通过非功能性需求约束和改进设计。

# 4. 结构视角

## 4.1 业务逻辑层的分解

业务逻辑层的开发包图见软件体系结构设计文档。

### 4.1.1 storagebl模块

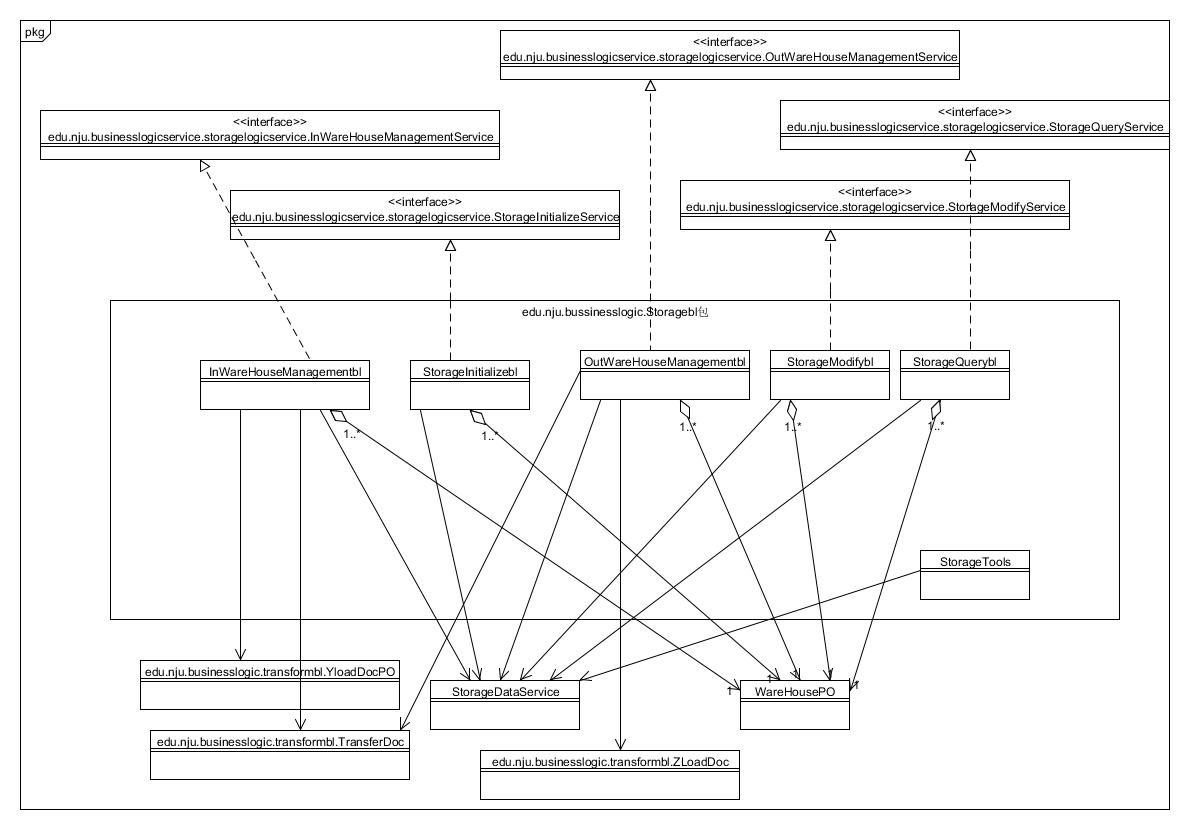
(1) 模块概述

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

Storagebl模块的职责及接口参见软件详细设计文档。

(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，我们加了Service层。比如逻辑层和展示层之间我们增加了businesslogicService层。Storagebl的模块设计如下所示：



Storagebl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| InWareHouseManagementbl | 负责生成入库单的VO |
| OutWareHouseManagementbl | 负责生成出库单的VO |
| StorageInitializebl | 负责库存初始化相关业务 |
| StorageModifybl | 负责库区调整的业务逻辑 |
| StorageQuerybl | 负责库存盘点的业务逻辑 |
| StorageTools | 负责为其他类提供工具服务 |

(3) 模块内部类的接口规范

InWareHouseManagementbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InWareHouseManagementbl.getVO\_Transfer | 语法 | public InWareHouseDocVO getInWareHouseDocVO\_Transfer(int TransferDocID) |
| 前置条件 | 已经有一个对应的中转单编号 |
| 后置条件 | 根据中转单编号生成入库单 |
| InWareHouseManagementbl.getVO\_YloadDocID | 语法 | public InWareHouseDocVO getInWareHouseDocVO\_YloadDoc(int YloadDocID) |
| 前置条件 | 已经有一个对应的营业厅装车单编号 |
| 后置条件 | 根据营业厅装车单编号生成入库单 |
| InWareHouseManagementbl.updataFile | 语法 | public void updateInWareHouseDoc(InWareHouseDocVO out) |
| 前置条件 | 已经生成了一个入库单 |
| 后置条件 | 更新入库单文件 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| TransferDocPO getTransferPO(int TransferID) | 根据中转单号获得中转单对象 | |
| WareHousePO getWareHouse() | 获得当前仓库的引用 | |
| public long getInWareHouseDocID(); | 获取当前入库单应得的编号 | |
| YloadDocPO getYloadDocPO(int YloadDocID) | 根据营业厅装车单号号获得营业厅装车单对象 | |
| updateInWareHouseDoc(InWareHouseDocPO out) | 根据一个入库单PO更新入库单文件 | |

OutWareHouseManagementbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OutWareHouseManagementbl.getVO\_ ZloadDoc | 语法 | public OutWareHouseDocVO getOutWareHouseDocVO\_ZloadDoc(int ZloadDocID) |
| 前置条件 | 已有一个对应的一个中转中心装车单编号 |
| 后置条件 | 生成出库单 |
| OutWareHouseManagementbl.getVO\_Transfer | 语法 | public OutWareHouseDocVO getOutWareHouseDocVO\_Transfer(int TransferDocID) |
| 前置条件 | 已有一个对应的中转单编号 |
| 后置条件 | 生成出库单 |
| OutWareHouseManagementbl.updataFile | 语法 | public void updateOutWareHouseDoc(OutWareHouseDocVO out) |
| 前置条件 | 已经生成了一个出库单 |
| 后置条件 | 更新出库单文件 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ZloadDoc getZloadDocPO(int ZloadDocID) | 根据中转中心装车单编号获得中转中心装车单对象 | |
| ArrayList<RecordPO> getStorageItemList(int[] SendDocIDList) | 根据快递编号列表，获得库存对象的应用列表 | |
| TransferPO getTransferPO(int TransferID) | 根据中转单编号获得中转单对象 | |
| void updateOutWareHouseDoc(OutWareHouseDocPO out) | 根据生成的出库单PO对象，更新文件 | |

StorageInitializebl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageInitializebl.clear | 语法 | public void clear() |
| 前置条件 | 无 |
| 后置条件 | 清除当前库存信息 |
| InWareHouseManagementbl.getVO\_YloadDocID | 语法 | public void addNewStorageItem(RecordPO recordPO) |
| 前置条件 | 已经输入了一条库存对象的各项数据 |
| 后置条件 | 更新了仓库的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public void clear(); | 将仓库文件的数据清空 | |
| public void addNewStorageItem(RecordPO recordPO) | 获得当前仓库的引用 | |
| public void addNewStorageItem(RecordPO recordPO) | 根据新添加的库存对象的数据更新库存文件 | |

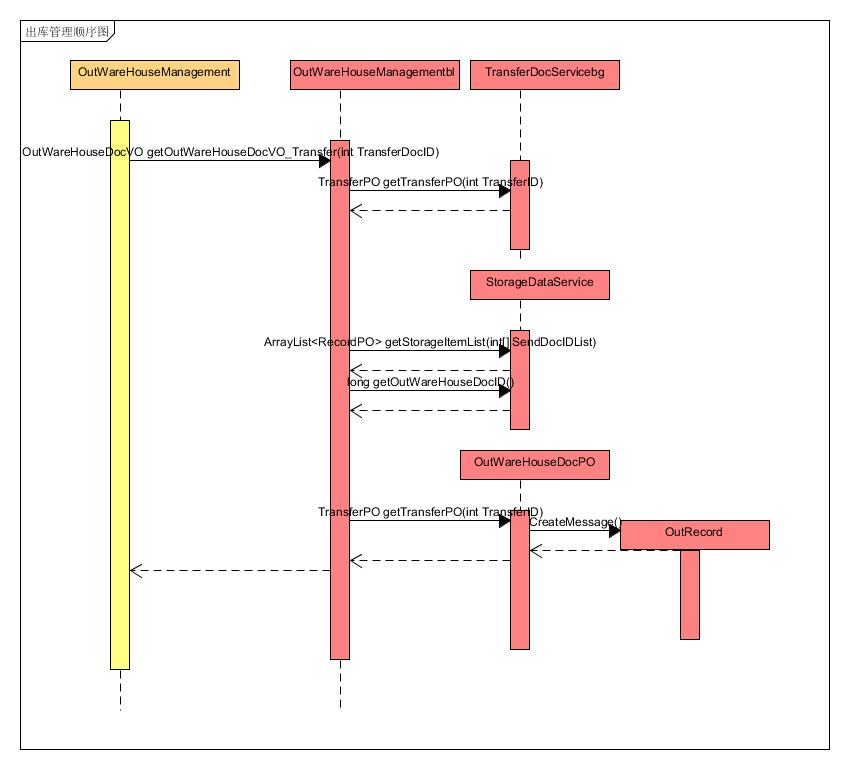
StorageQuerybl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageQuerybl.getInWareHouseDocVO\_Fly() | 语法 | public ArrayList<RecordPO> getInWareHouseDocVO\_Fly() |
| 前置条件 | 无 |
| 后置条件 | 获得当前仓库的航运区的库存对象列表 |
| StorageQuerybl.getInWareHouseDocVO\_Train() | 语法 | public ArrayList<RecordPO> getInWareHouseDocVO\_Train() |
| 前置条件 | 无 |
| 后置条件 | 获得当前仓库的货运区的库存对象列表 |
| StorageQuerybl. getInWareHouseDocVO\_Car() | 语法 | public ArrayList<RecordPO> getInWareHouseDocVO\_Car() |
| 前置条件 | 无 |
| 后置条件 | 获得当前仓库的汽运区的库存对象列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public ArrayList<RecordPO> getInWareHouseDocVO\_Fly() | 数据层需要读取文件，生成航运区的库存对象list传给逻辑层 | |
| public ArrayList<RecordPO> getInWareHouseDocVO\_Train() | 数据层需要读取文件，生成货运区的库存对象list传给逻辑层 | |
| public ArrayList<RecordPO> getInWareHouseDocVO\_Car() | 数据层需要读取文件，生成汽运区的库存对象list传给逻辑层 | |

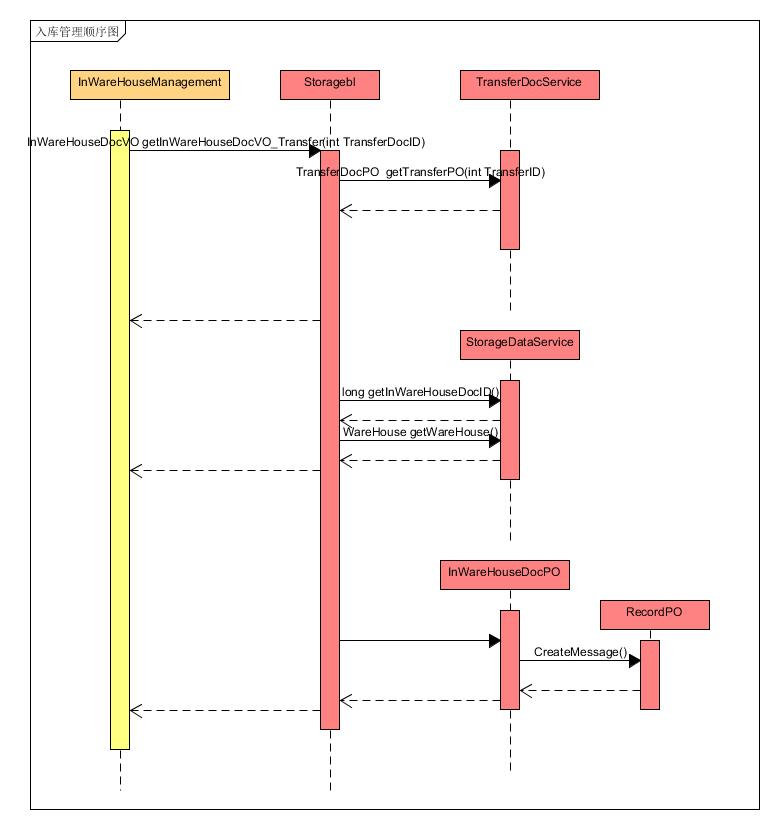
StorageModifybl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageModifybl.setAlarm | 语法 | public void setAlarm(double p) |
| 前置条件 | 已经输入一个预警阈值 |
| 后置条件 | 更新了文件里的预警阈值 |
| StorageModifybl.setPercent | 语法 | public void setPercent(double p) |
| 前置条件 | 已经输入了一个调整机动区到报警区的百分比 |
| 后置条件 | 调整了对应百分比的机动区空间到报警区，修改了仓库的属性 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public void setAlarm(double p) | 根据逻辑层的预警阈值更新文件中的持久化的值 | |
| public void setPercent(double p) | 更新文件 | |

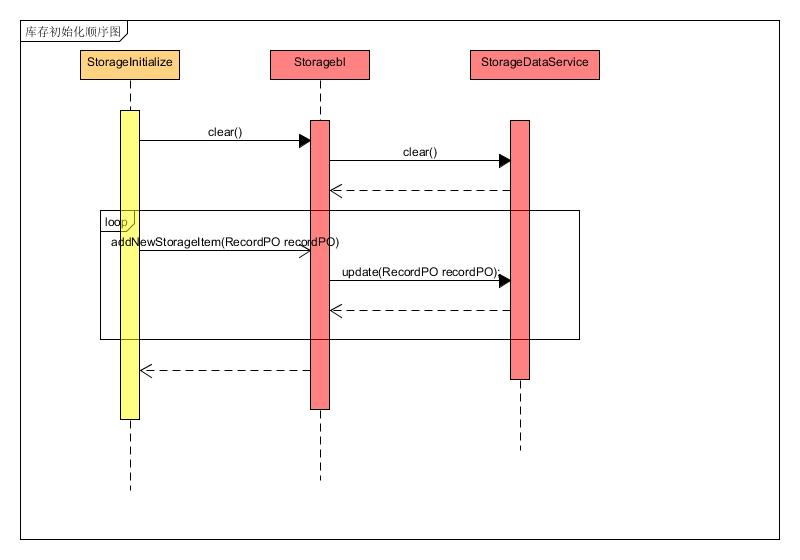
(4) 业务逻辑层的动态模型



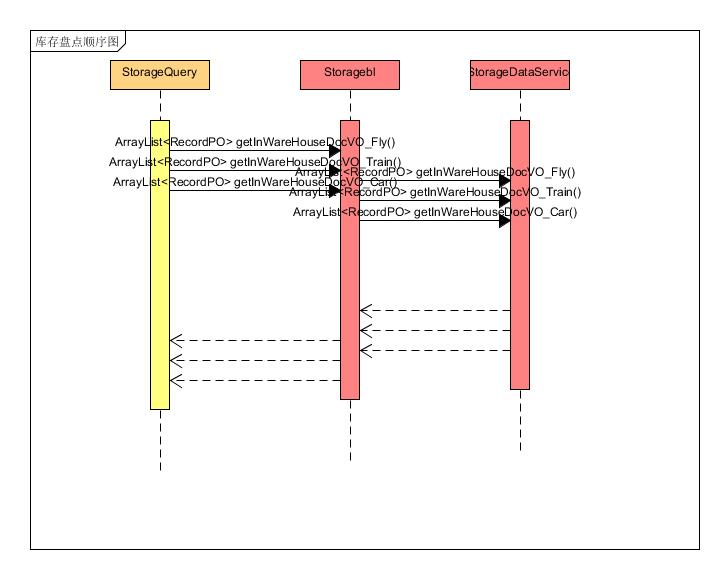
出库管理顺序图



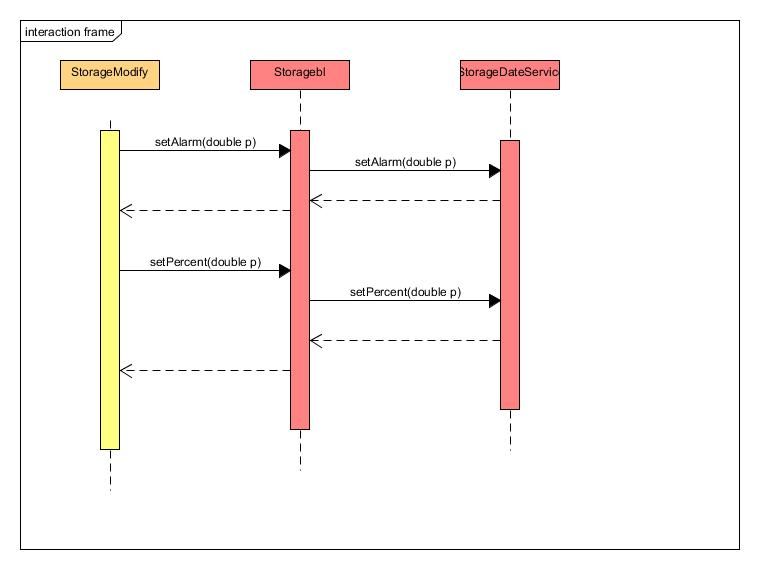
入库管理顺序图



库存初始化顺序图



库存盘点顺序图



库区调整顺序图

(5) 业务逻辑层的设计原理

采用分散式风格，每个逻辑模块内部类只负责一个功能流程。互相尽少干涉。

### 4.1.2 logispicsquerybl模块

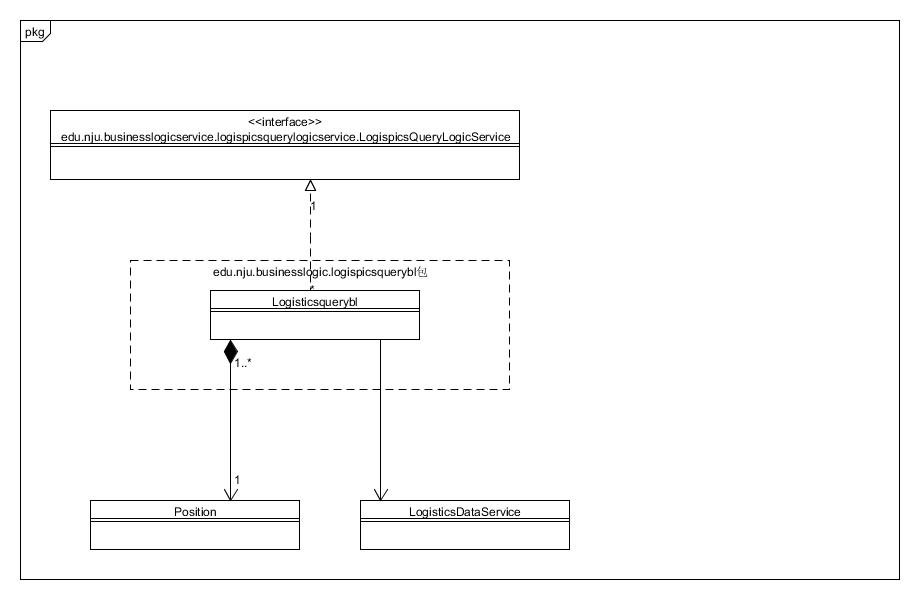
(1) 模块概述

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

Logispicsquerybl模块的职责及接口参见软件系统结构描述文档。

(2) 整体结构

物流查询这个模块分为展示层，业务逻辑层，数据层。展示层与逻辑层之间有logispicsqueryblService接口，逻辑层与数据层之间有LogisticsDataService接口。PositionPO是作为快递历史轨迹的持久化对象被添加到设计模型中去的。

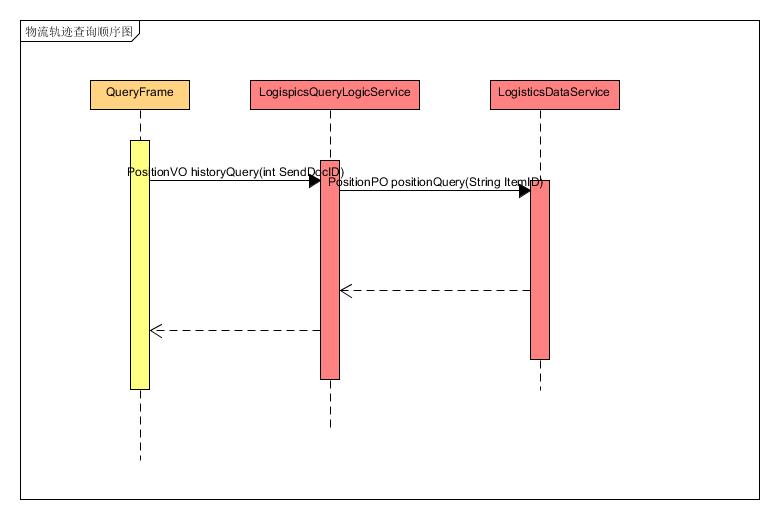


(3) 模块内部类的接口规范

Logispicsquerybl类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Logispicsquerybl. HistoryQuery | 语法 | public PositionVO historyQuery(int SendDocID) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Logispicsquerybl.updatePosition | 语法 | public void changePosition(long ItemID,String pos) |
| 前置条件 | 无 |
| 后置条件 | 更新某快递的历史轨迹信息 |
| Logispicsquerybl.addPosition | 语法 | public void createPosition(long Item,String pos) |
| 前置条件 | 无 |
| 后置条件 | 增加某个快件的历史轨迹信息记录 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public void changePosition(long ItemID,String pos); | 更新快递号为Item的快递的历史轨迹信息 | |
| public void createPosition(long Item,String pos); | 增加快递号为Item的快递的历史轨迹记录 | |
| PositionPO positionQuery(String ItemID); | 查询某快递号的快递的历史轨迹信息 | |

(4) 业务逻辑层的动态模型



物流轨迹查询顺序图

### collectionbl模块

1. 模块概述

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

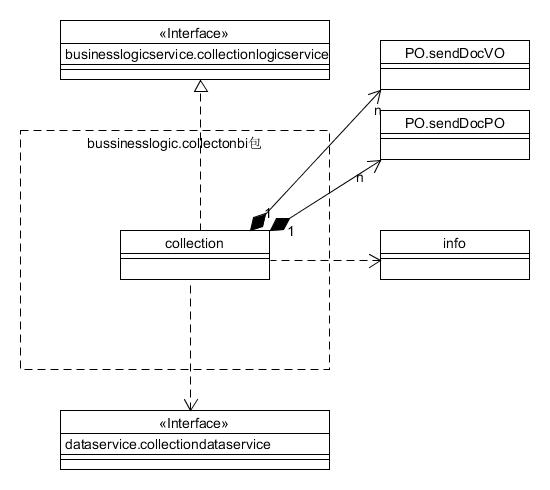
Collectionbl模块的职责及接口参见软件体系结构描述文档

(2) 整体结构

展示层、业务逻辑层、数据层间由collectionLogicService接口与collectionDataService分隔，业务逻辑完全由collection对象完成，sendDocPO是寄件单的持久化对象，sendDocVO

是寄件单的可视化对象。

Collectionbl模块设计如图所示

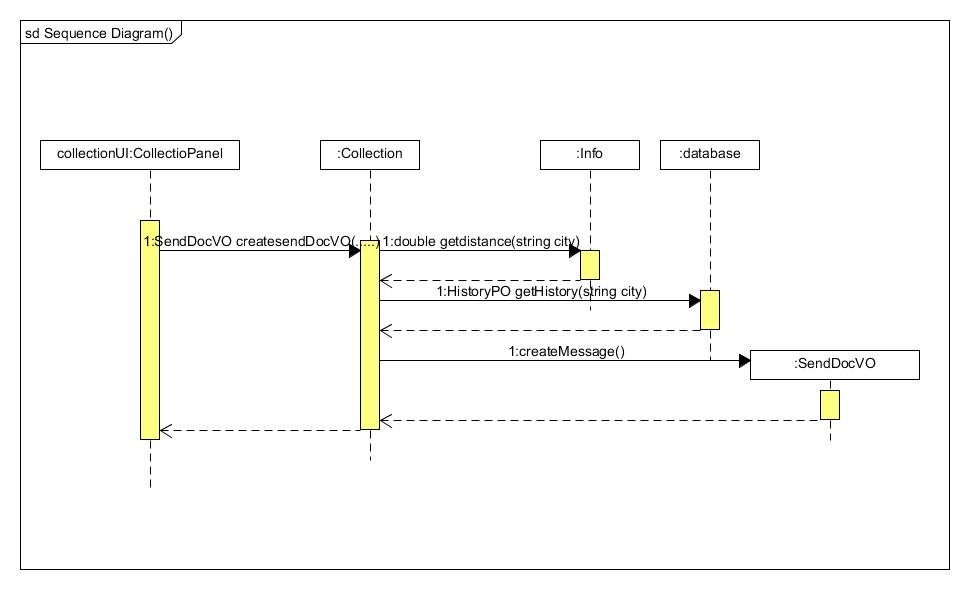


(3) 模块内部类的接口规范

Collection的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| Collection.QueryGoodsInfo | 语法 | | Public List<PositionPO> QueryGoodsInfo(int SendDocID) |
| 前置条件 | | SendDocID 正确 |
| 后置条件 | | 根据寄件单编号查询物流状态信息，返回一个List<PositionPO> |
| Collection.getUncheckedSendDoc | 语法 | | Public list<SendDocPO> getAllSendDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 得到待审批寄件单 |
| Collection.getCourierMoney(Courier courier) | 语法 | | Public Int getCourierMoney（Courier courier） |
| 前置条件 | | 快递员正确 |
| 后置条件 | | 得到该快递员的收款金额（通过寄件单文件） |
| Collectionbl.getSendDocIDList(Courier courier) | 语法 | | Public list<int> getSendDocIDList(Courier courier) |
| 前置条件 | | 快递员正确 |
| 后置条件 | | 得到快递员的订单条形码号（通过寄件单文件） |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| CollectionDataService.saveSendDocPO(SendDocPO po) | | 保存寄件单单一持久化对象 | |
| CollectionDataService.getDistance(String city) | | 获取距离数据 | |
| CollectionDataService.getSequence() | | 获取序列号 | |
| CollectionDataService.changeSequence(int sequence) | | 改变最后序列号 | |
| CollectionDataService.saveHistory(HistoryPO po) | | 保存历史数据单一持久化对象 | |

(4) 业务逻辑层的动态模型

图示表明了系统中，快递员输入寄件单相关信息后，揽件逻辑处理的相关对象之间的协作

(5) 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由不同的逻辑对象提供

### 4.1.4 transferbl模块

1. 模块概述

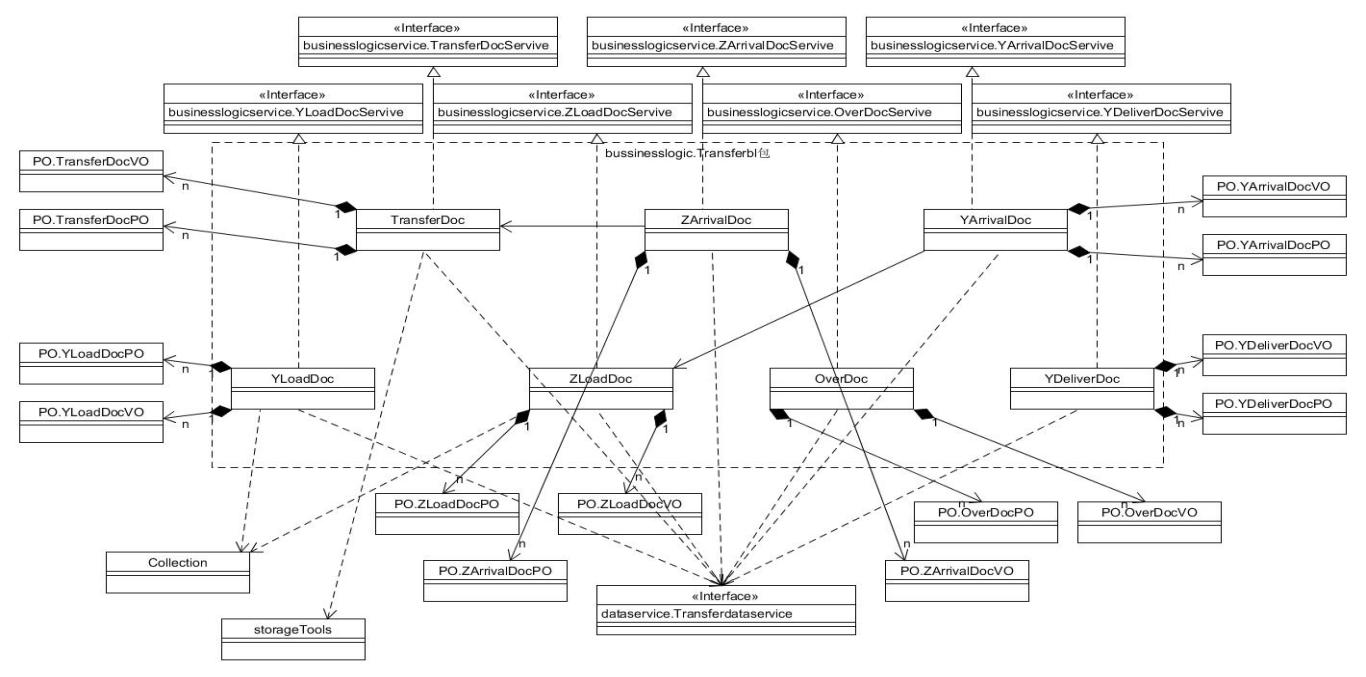
Transferbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

Transferbl模块的职责及接口参见软件体系结构描述文档

1. 整体结构

展示层、业务逻辑层、数据层间由TransferLogicService接口与TransferDataService分隔，业务逻辑根据不同操作对象，由TransferDoc、ZArrivalDoc等对象完成，TransferDocPO、ZArrivalDocPO等是各单据的持久化对象，TransferDocVO、ZArrivalVO等是各单据的可视化对象。

Transferbl模块设计如图所示



1. 模块内部类的接口规范

TransferDoc的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 供接口 | | | |
| TransferDoc.getUncheckedTransferDoc | 语法 | | Public TransferDocPO getAllTransferDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 得到待审批中转单 |
|  | | | |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| TransferDataService.saveTransferDocPO(TransferDocPO po) | | 保存中转单单一持久化对象 | |
| TransferDataService.getTransferSequence() | | 获取中转单序列号 | |
| TransferDataService.changeTransferSequence(int TransferSequence) | | 改变中转单最后序列号 | |
| TransferDataService.getTransferDocPO(int TransferDocID) | | 根据ID查找中转单单一持久化对象 | |
|  | |  | |

YDeliverDoc的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 供接口 | | | |
| YDeliverDoc.getUncheckedYDeliverDoc | 语法 | | Public YDeliverDocPO DegetYDeliverDoc() |
| 前置条件 | | 得到派件单 |
| 后置条件 | | 无 |
|  | | | |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| TransferDataService.saveYDeliverDocPO(YDeliverDocPO po) | | 保存派件单单一持久化对象 | |
| TransferDataService.getYDeliverSequence() | | 获取派件单序列号 | |
| TransferDataService.changeYDeliverSequence(int YDeliverSequence) | | 改变派件单最后序列号 | |
| TransferDataService.getYDeliverDocPO(int DeliverDocID) | | 根据ID查找派件单单一持久化对象 | |
|  | |  | |

OverDoc的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 供接口 | | | |
| OverDoc.getUncheckedOverDoc | 语法 | | Public OverDocPO getUncheckedOverDoc() |
| 前置条件 | | 得到收件单 |
| 后置条件 | | 无 |
|  | | | |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| TransferDataService.saveOverDocPO(OverDocPO po) | | 保存收件单单一持久化对象 | |
| TransferDataService.getOverSequence() | | 获取收件单序列号 | |
| TransferDataService.changeOverSequence(int OverSequence) | | 改变收件单最后序列号 | |
| TransferDataService.getOverDocPO(int OverDocID) | | 根据ID查找收件单单一持久化对象 | |
|  | |  | |

YLoadDoc的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 供接口 | | | |
| YLoadDoc.getUncheckedYloadDoc | 语法 | | Public YloadDocPO getAllYloadDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 得到待审批营业厅装车单 |
|  | | | |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| TransferDataService.saveYloadDocPO(ZloadDocPO po) | | 保存装车单单一持久化对象 | |
| TransferDataService.getYloadSequence() | | 获取装车单序列号 | |
| TransferDataService.changeYloadSequence(int YloadSequence) | | 改变装车单最后序列号 | |
| TransferDataService.getYloadDocPO(int YloadDocID) | | 根据ID查找装车单单一持久化对象 | |
|  | |  | |

ZLoadDoc的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 供接口 | | | |
| ZLoadDoc.getUncheckedZloadDoc | 语法 | | Public ZloadDocPO getAllZloadDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 得到待审批营业厅装车单 |
|  | | | |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| TransferDataService.saveZloadDocPO(ZloadDocPO po) | | 保存装车单单一持久化对象 | |
| TransferDataService.getZloadSequence() | | 获取装车单序列号 | |
| TransferDataService.changeZloadSequence(int ZloadSequence) | | 改变装车单最后序列号 | |
| TransferDataService.getZloadDocPO(int ZloadDocID) | | 根据ID查找装车单单一持久化对象 | |
|  | |  | |

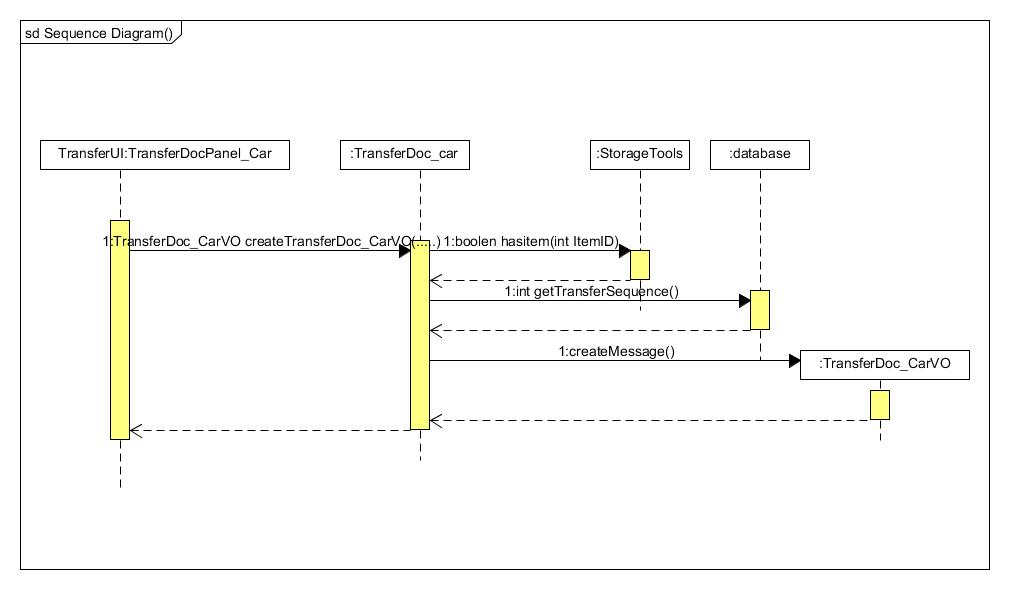
YArrivalDoc的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 供接口 | | | |
| YArrivalDoc.getUncheckedYArrivalDoc | 语法 | | Public YArrivalDocPO getAllYArrivalDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 得到待审批到达单 |
|  | | | |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| TransferDataService.saveYArrivalDocPO(YArrivalDocPO po) | | 保存营业厅到达单单一持久化对象 | |
| TransferDataService.getYArrivalSequence() | | 获取营业厅到达单序列号 | |
| TransferDataService.changeYArrivalSequence(int YArrivalSequence) | | 改变营业厅到达单最后序列号 | |
| TransferDataService.getYArrivalDocPO(int YArrivalDocID) | | 根据ID查找营业厅到达单单一持久化对象 | |
|  | |  | |

ZArrivalDoc的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 供接口 | | | |
| ZArrivalDoc.getUncheckedZArrivalDoc | 语法 | | Public ZArrivalDocPO getAllZArrivalDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 得到待审批中转中心到达单 |
|  | | | |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| TransferDataService.saveZArrivalDocPO(ZArrivalDocPO po) | | 保存中转中心到达单单一持久化对象 | |
| TransferDataService.getZArrivalSequence() | | 获取中转中心到达单序列号 | |
| TransferDataService.changeZArrivalSequence(int ZArrivalSequence) | | 改变中转中心到达单最后序列号 | |
| TransferDataService.getZArrivalDocPO(int ZArrivalDocID) | | 根据ID查找中转中心到达单单一持久化对象 | |
|  | |  | |

1. 业务逻辑层的动态模型

图示表明了系统中，中转中心业务员选择汽运中转，输入到达地，车次号，监装员和所有单号后，揽件逻辑处理的相关对象之间的协作

1. 业务逻辑层的设计原理

利用分散式控制风格，每个操作不同对象的业务逻辑由不同的逻辑对象提供

### 4.1.5 approvelbl模块

### 4.1.6 financebl模块

### 4.1.7 infobl模块

（1）模块概述

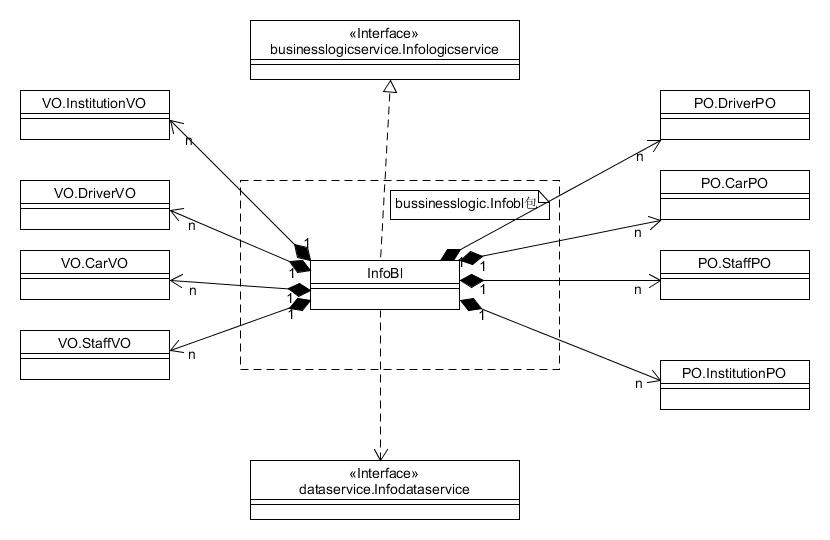
InfoBl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

InfoBl模块的职责及接口参见软件系统结构描述文档相关表格。

（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，我们添加了接口。业务逻辑层和展示层之间有businesslogicservice.InfoBlService接口，业务逻辑层和数据层之间添加dataservice.InfoDataService接口。DriverPO，CarPO，InstitutionPO，StaffPO分别是对应的持久化对象。

模块的设计模式如图

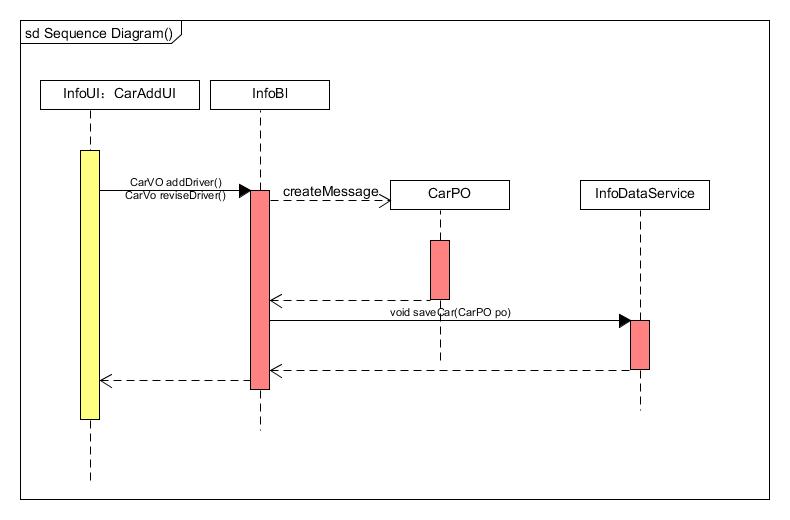


（3）模块内部类的接口规范

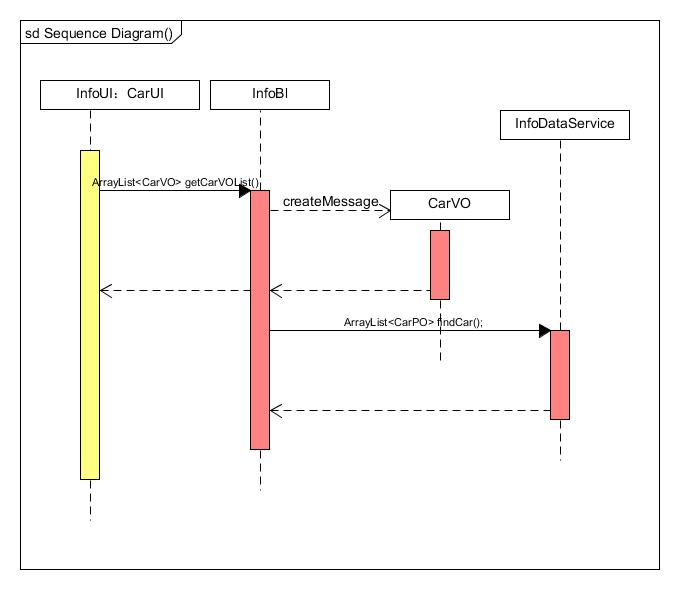
InfoBl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InfoBl.getDriveVOList | 语法 | public ArrayList<DriverVO> getDriveVOList(); |
| 前置条件 | 进入driverUI |
| 后置条件 | 获得driverVOList |
| InfoBl.getCarVOList | 语法 | public ArrayList<CarVO> getCarList(); |
| 前置条件 | 进入CarUI |
| 后置条件 | 获得CarList |
| InfoBl. getSalary | 语法 | public int getSalary(int id); |
| 前置条件 | 进入修改薪资界面 |
| 后置条件 | 获得相应的工资 |
| InfoBl. getStaffVOList | 语法 | public ArrayList<StaffVO> getStaffVOList()； |
| 前置条件 | 进入人员管理界面 |
| 后置条件 | 获得相应的VOList |
| InfoBl. getInstitutionVOList | 语法 | public ArrayList<InstitutionVO> getInstitutionVOList()； |
| 前置条件 | 进入机构管理界面 |
| 后置条件 | 获得相应的VOList |
| InfoBl. getDistance | 语法 | public double getDistance(String city1,String city2)； |
| 前置条件 | 进入距离修改界面 |
| 后置条件 | 获得城市间距离 |
| InfoBl. saveDriver | 语法 | public void saveDriver(DriverPO po)； |
| 前置条件 | 修改Driver的信息 |
| 后置条件 | 保存driverPO |
| InfoBl. saveCar | 语法 | public void saveCar(CarPO po); |
| 前置条件 | 修改车辆信息 |
| 后置条件 | 保存carPO |
| InfoBl. saveSalary | 语法 | public void saveSalary (StaffPO po); |
| 前置条件 | 修改薪水信息 |
| 后置条件 | 保存对应的staffPO |
| InfoBl. saveStaff | 语法 | public void saveStaff(StaffPO po); |
| 前置条件 | 修改人员信息 |
| 后置条件 | 保存对应的staffPO |
| InfoBl. saveInstitution | 语法 | public void saveInstitution (InstitutionPO po); |
| 前置条件 | 修改机构信息 |
| 后置条件 | 保存对应的InstitutionPO |
| InfoBl. saveDistance | 语法 | public void saveDistance(String city,String city2,double distance); |
| 前置条件 | 修改城市距离 |
| 后置条件 | 保存修改的距离参数 |
| InfoBl. getDistance | 语法 | public double getDistance(String city); |
| 前置条件 | 快递员揽件 |
| 后置条件 | 获得该快递员所在城市和目的地的距离 |

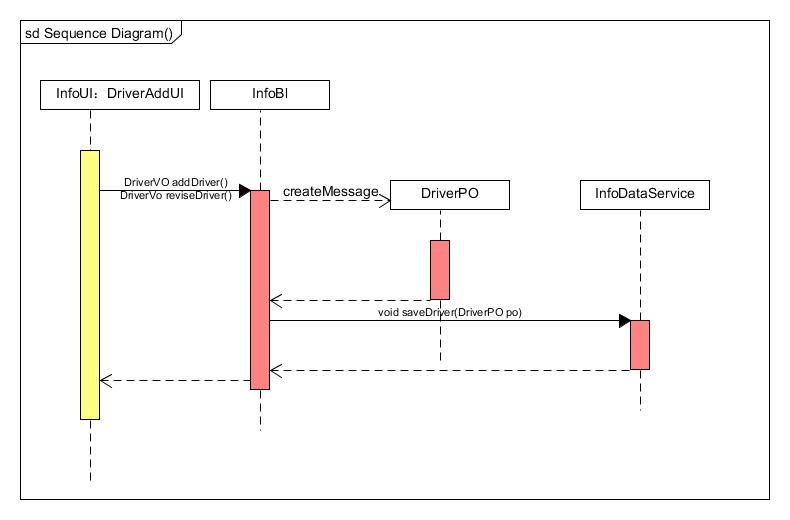
（4）业务逻辑层的动态模型



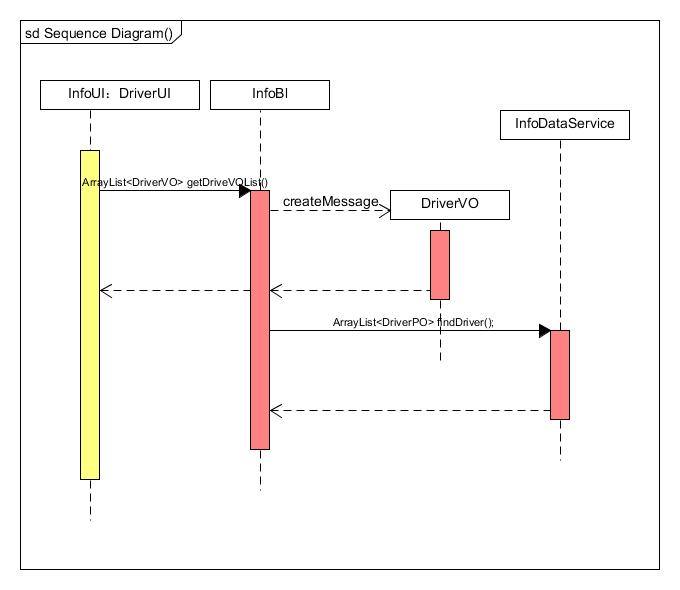
增加和修改车辆信息的顺序图



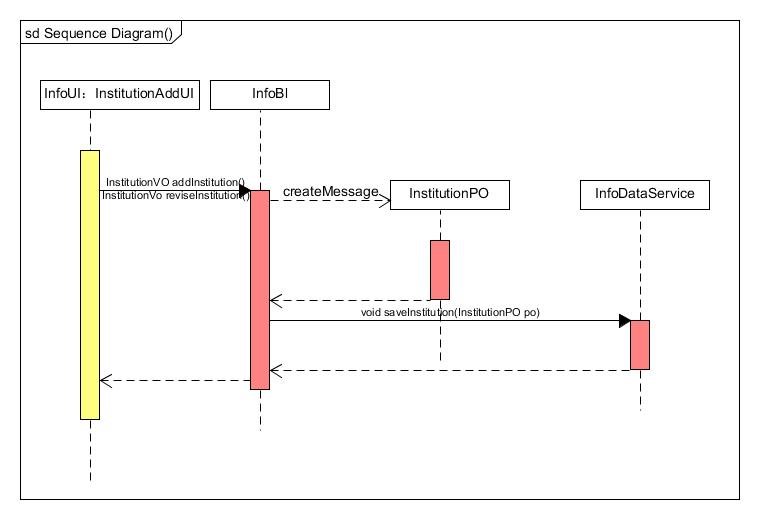
查询车辆信息的顺序图



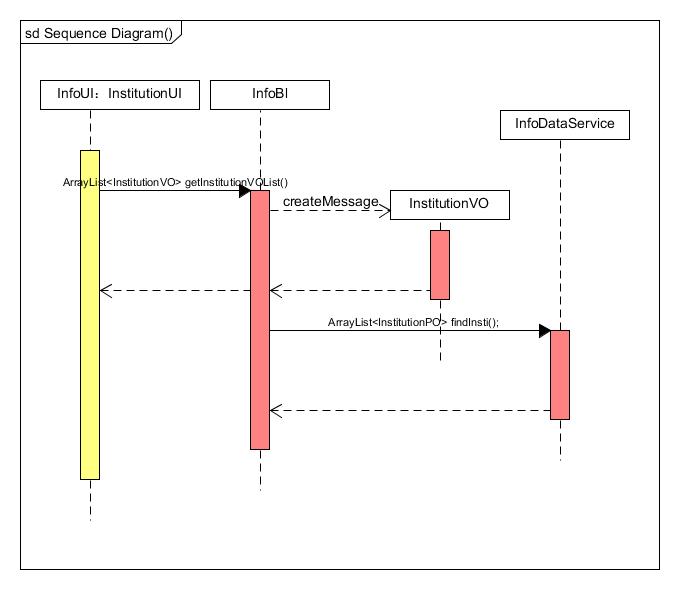
增加和修改司机信息的顺序图



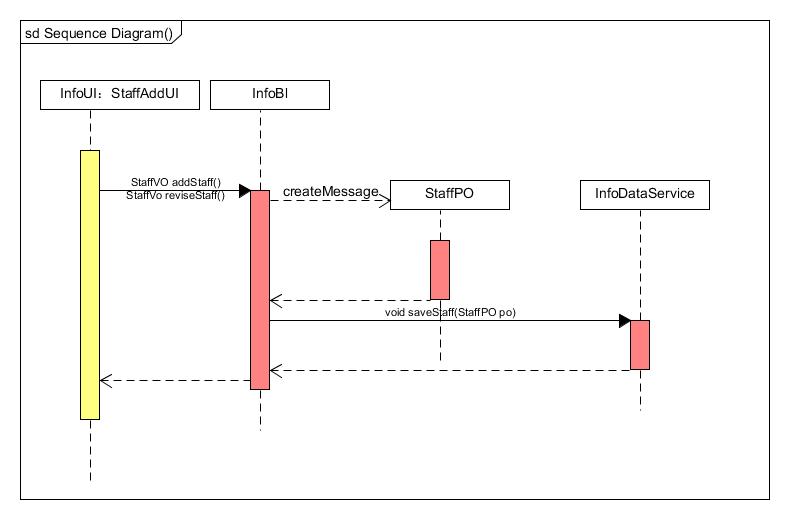
司机信息查看的顺序图



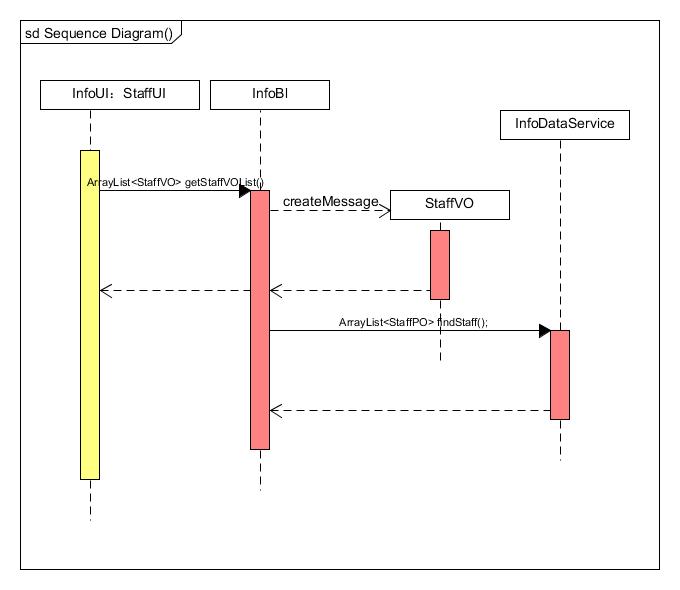
增加修改机构的顺序图



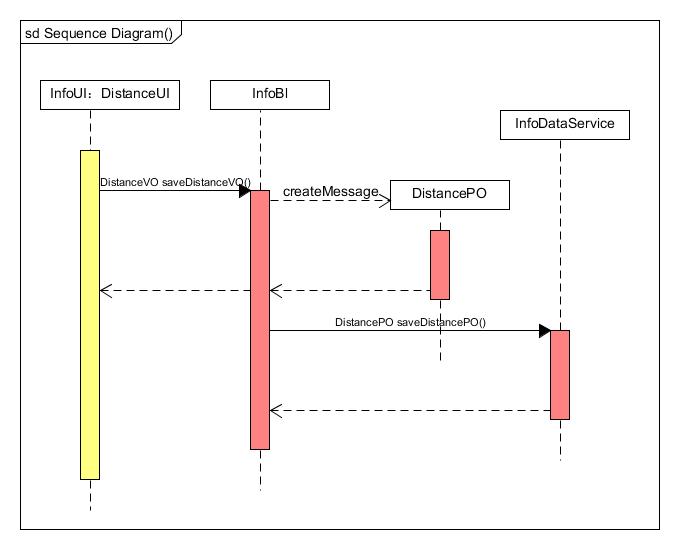
查看机构信息的顺序图



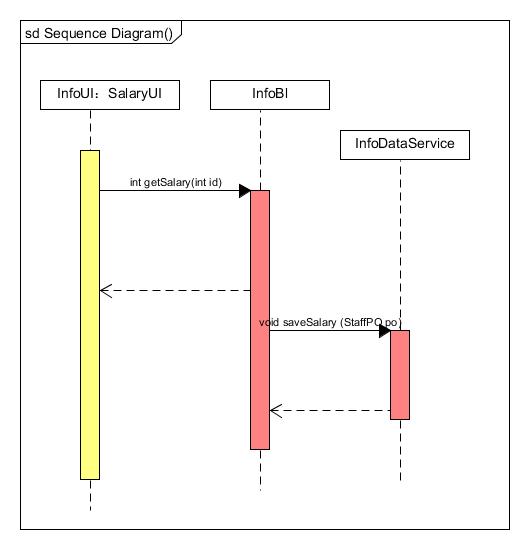
增加和修改人员的顺序图



查看人员信息的顺序图



距离修改的顺序图



薪水策略修改的顺序图

（5）业务逻辑层的设计原理

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

### 4.1.8 systembl模块

（1）模块概述

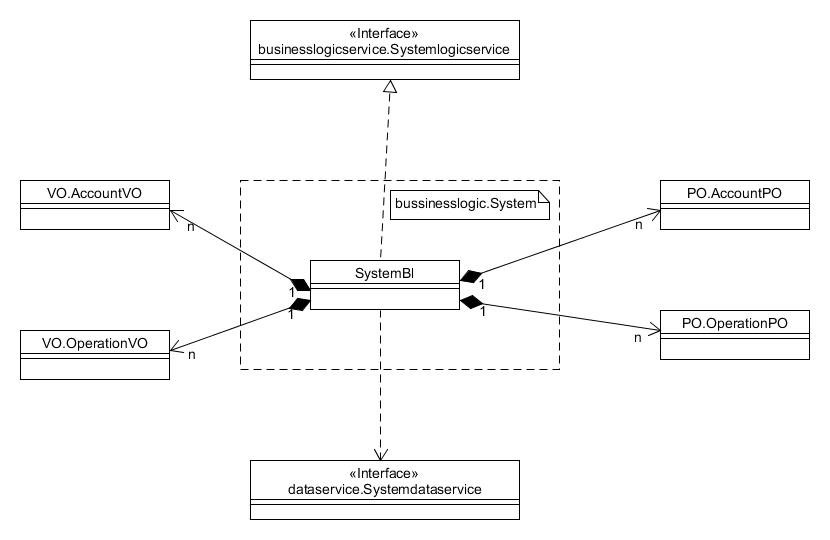
SystemBl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

SystemBl模块的职责及接口参见软件系统结构描述文档相关表格。

（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，我们添加了接口。业务逻辑层和展示层之间有businesslogicservice. SystemBlService接口，业务逻辑层和数据层之间添加dataserviceSystemDataService接口。OperationPO,AccountPO分别是对应的持久化对象。

模块的设计如图所示。

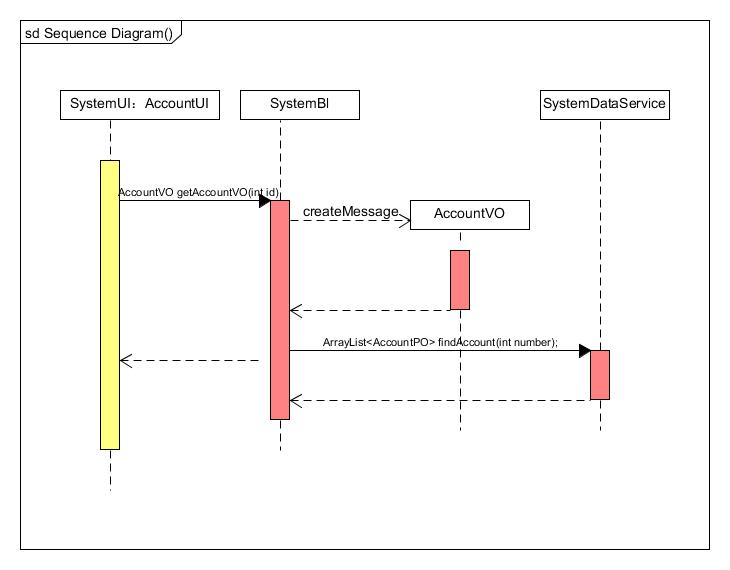


（3）模块内部类的接口规范

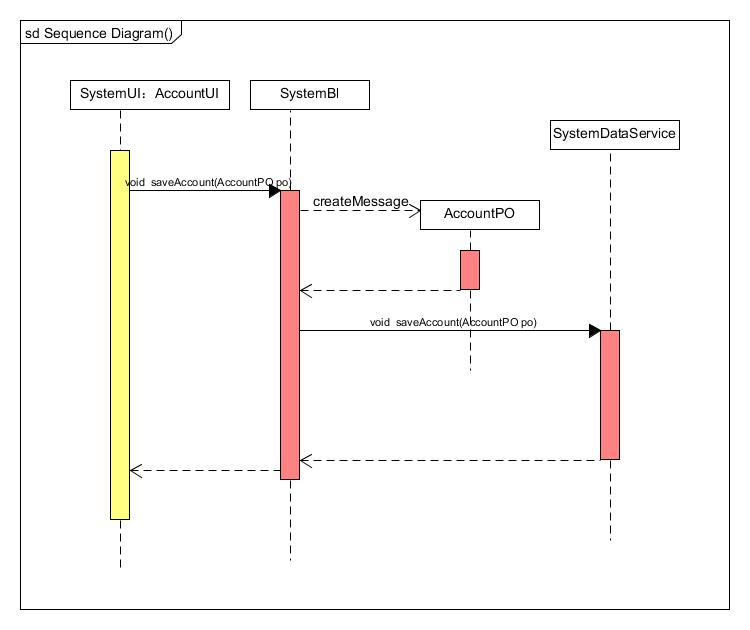
SystemBl接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SystemBl. getOperationVOList | 语法 | public ArrayList<OperationVO> getOperationVOList(long beginTime,long endTime); |
| 前置条件 | 进入操作日志查询界面 |
| 后置条件 | 获得操作日志VO |
| SystemBl. getAccountVO | 语法 | public AccountVO getAccountVO(int id); |
| 前置条件 | 进入账户权限管理界面 |
| 后置条件 | 获得账户权限信息 |
| SystemBl. saveAccount | 语法 | public void saveAccount(AccountPO po); |
| 前置条件 | 修改账户权限 |
| 后置条件 | 保存对应的账户PO |

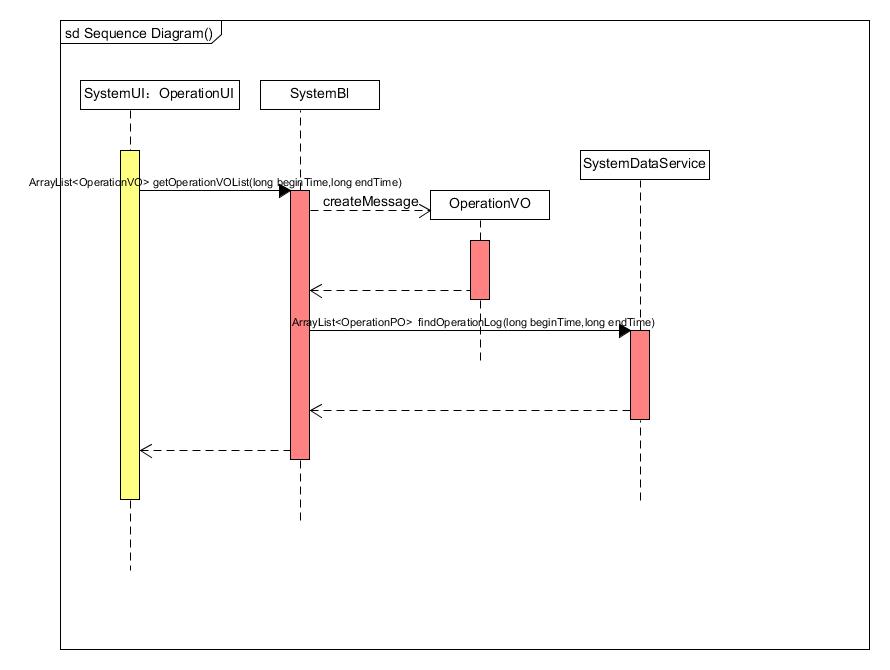
（4）业务逻辑层的动态模型



查看账户权限的顺序图



修改账户权限的顺序图



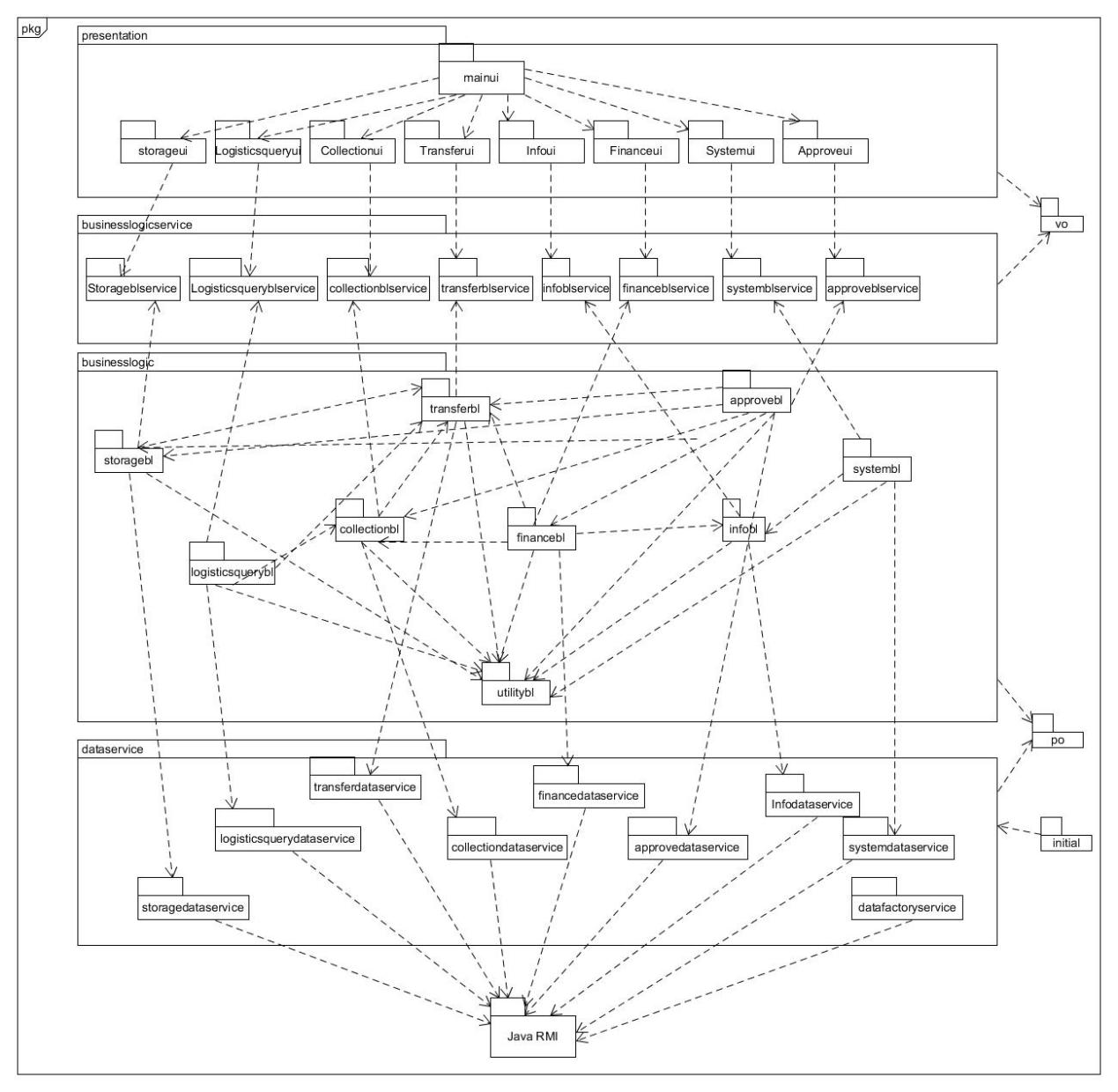
查看系统日志的顺序图

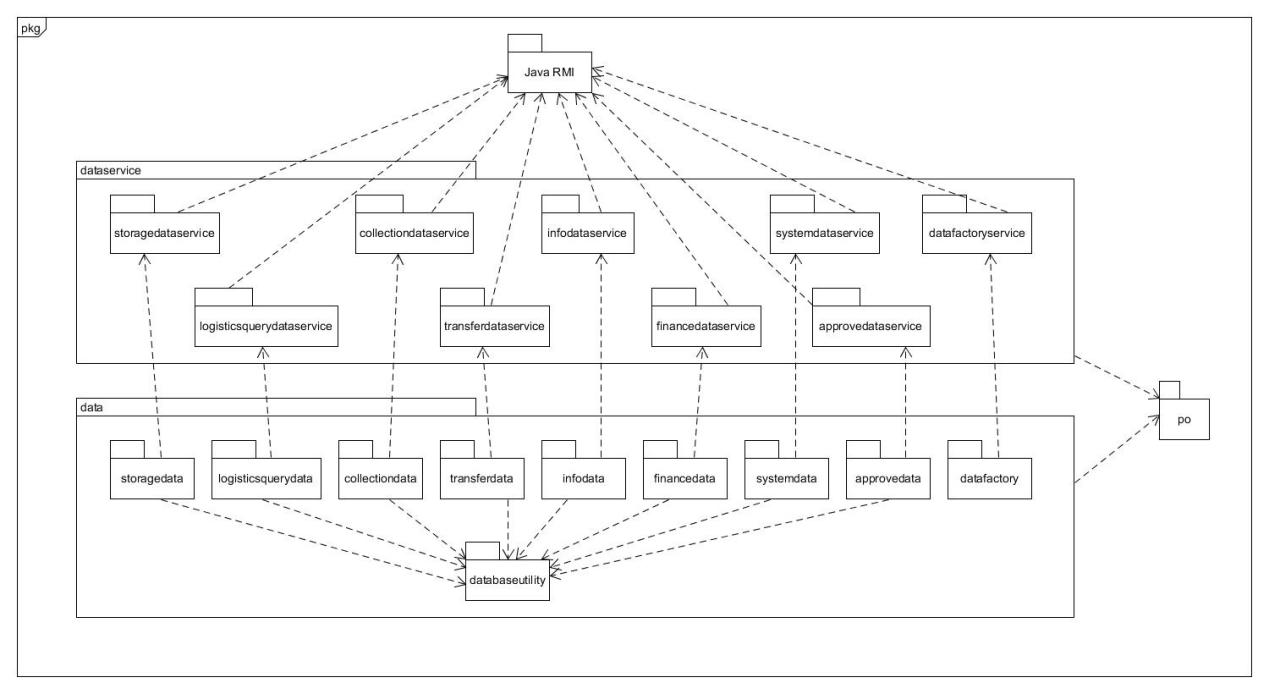
（5）业务逻辑层的设计原理

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

# 5. 依赖视角

客户端和服务器端各自包之间的依赖关系如下图所示。

 客户端开发包图



服务器开发包图