详细设计描述文档

快递物流系统软件

学校： 南京大学

院系： 软件学院

编写小组： 33

目录

1 引言

1.1编制目的

本文档承接快递物流系统软件体系描述文档，继续完成快递物流系统的详细设计工作，达到指导后续的软件构造的目的，同时实现设计人员与构造人员、测试人员、用户之间的交流。

本文档的预期读者为开发人员、构造人员、测试人员、最终用户以及项目管理人员。

1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统 | 为本次工程的开发产品 |
| vo | 值对象 | 用于业务层之间的数据传递 |
| po | 持久对象 | 与数据库中的表对应的Java对象 |

1.3导读

该报告分为5个部分，第一部分是引言，主要是介绍文档的基本信息；第二部分是产品概述，是快递物流系统的概括性描述；第三部分是对体系结构设计的概述；第四、第五和第六部分分别是结构视角、依赖视角，从多个视角对软件设计进行详细的描述。

1.4文档约定

对文档的的任何改动都应经过质量保证人员的统一，详细改动由小组成员共同商议。

1.5感谢

本用例文档由小组成员黄迪璇编写，何永俊负责审阅，详细描述由何永俊、许玥琪、邓逸鹏和黄迪璇共同完成，感谢他们的辛勤付出。

1.6参考资料

《软件工程与计算（卷二）——软件开发的技术基础》 机械工程出版社

《快递物流系统软件需求规格说明文档v0.5》

《快递物流系统软件体系描述文档v0.8》

2 产品概述

快递物流系统是为一本地民营物流企业开发的业务系统，开发的目的是帮助该企业处理日常的重点业务，包括收发快件，库存管理，财务操作和人员机构管理。

通过快递物流系统的应用，期望系统上线运行六个月后，保持合理库存，提高用户服务体验，增加业务额，提高财务人员工作效率，为经理的决策做支持。

3 体系结构设计概述

参考《快递物流系统软件体系描述文档v0.8》中对体系结构设计的概述。

4 结构视角

4.1pamanagementbl模块

(1)模块描述

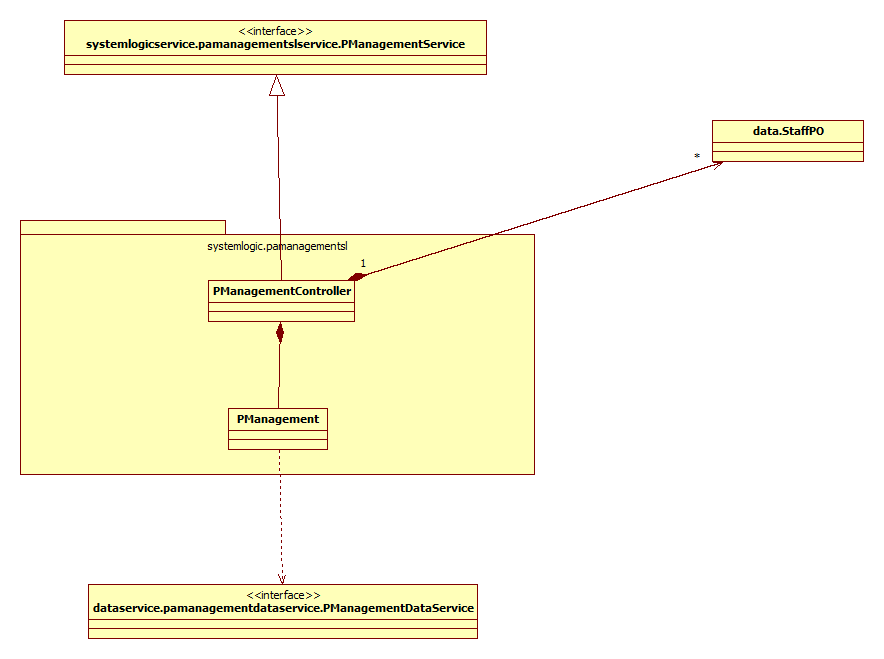
pamanagementbl模块承担的需求参见快递物流系统需求规格说明文档功能需求及相关非功能需求。

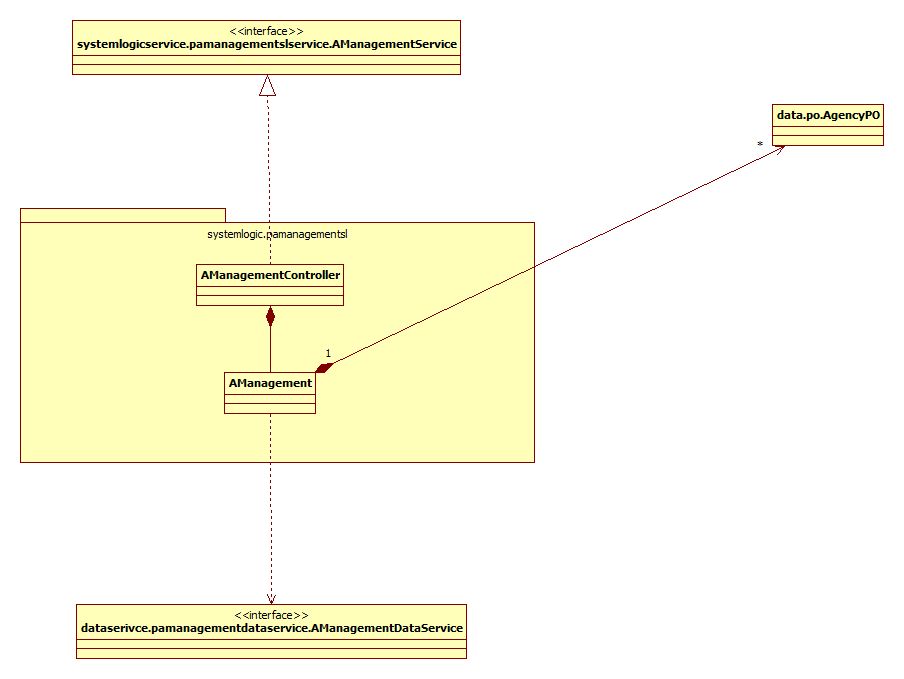
pamanagementbl模块的职责及接口参见快递物流系统软件体系描述文档。

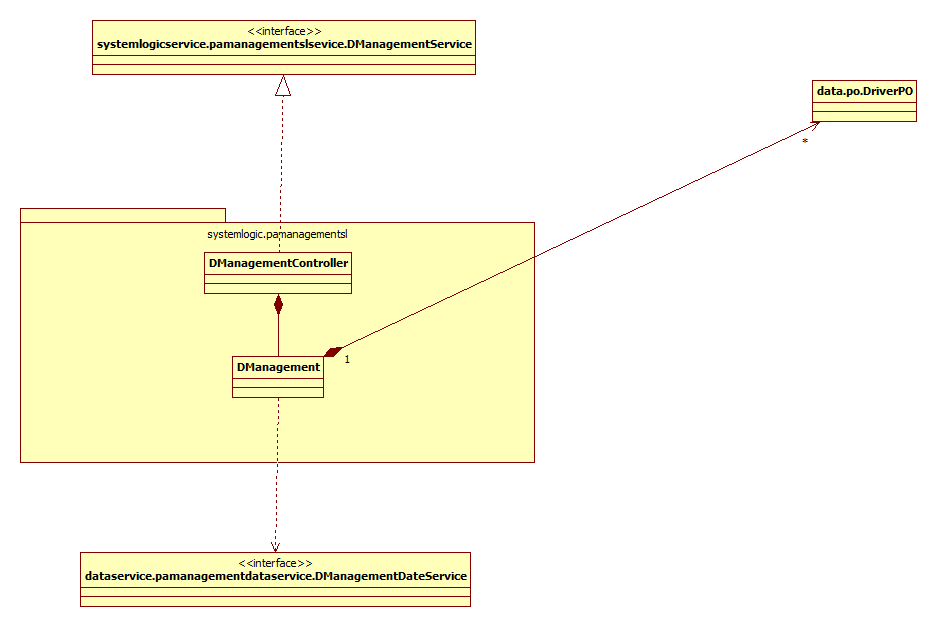
(2)整体结构

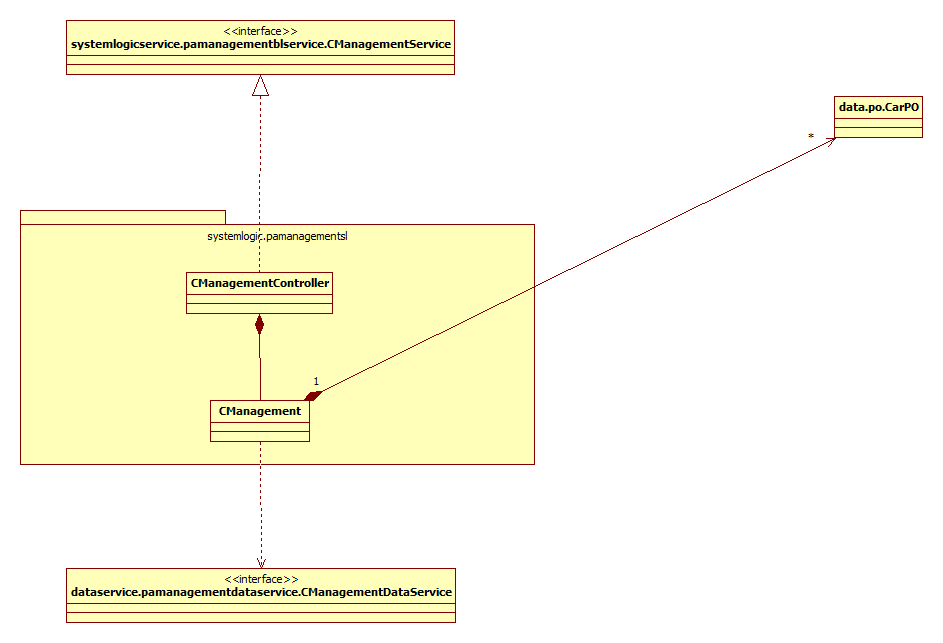
根据体系结构的设计，采用分层风格，将系统分为展示层、系统逻辑层、数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了pamanagementblservice和pamanagementdataservice两个接口。为了隔离系统逻辑职责和逻辑控制职责，我们添加了PManagementController、AManagementController、DManagementController、CManagementController，这样它们会将对人员机构管理的系统逻辑处理委托给PManagement、AMangement、DManagement、CManagement对象。AgencyPO、StaffPO、DriverPO、CarPO是作为人员机构管理信息的持久化对象被添加到设计模型上的。

pamanagementsl模块的图如下图所示。









pamanagementsl模块各个类的职责如下表

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应登录界面所需要的服务 |
| PManagementController | 负责实现人员管理界面所需要的服务 |
| AManagementController | 负责实现机构管理界面所需要的服务 |
| DManagementController | 负责实现司机管理界面所需要的服务 |
| CManagementController | 负责实现车辆管理界面所需要的服务 |
| PManagement | 人员管理的领域模型对象，拥有一次人员管理所持有的人员信息单列表、操作记录等信息，可以帮助完成人员管理界面说需要的服务 |
| AManagement | 机构管理的领域模型对象，拥有一次机构管理所持有的机构信息单列表、操作记录等信息，可以帮助完成机构管理界面所需要的服务 |
| DManagement | 司机管理的领域模型对象，拥有一次司机管理所持有的司机信息单列表、操作记录等信息，可以帮助完成司机管理界面所需要的服务 |
| CManagement | 车辆管理的领域模型对象，拥有一次车辆管理所持有的车辆信息单列表、操作记录等信息，可以帮助完成车辆管理界面所需要的服务 |

(3)模块内部类的接口规范

PManagementController和PManagement的接口规范如下表

PManagementController的接口规范

|  |  |  |
| --- | --- | --- |
| PManagementController.select | 语法 | public StaffVO select(String id); |
| 前置条件 | 已开启新的一轮人员管理 |
| 后置条件 | 调用PManagement领域对象的select方法 |
| PManagementController.delect | 语法 | public void delect(String id); |
| 前置条件 | 在一轮的人员信息管理，已选择好需要删除的人员信息单 |
| 后置条件 | 调用PManagement领域对像的delect方法 |
| PManagementContorller.revise | 语法 | public StaffVO revise(String id); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被选择 |
| 后置条件 | 调用PManagement领域对象的revise方法 |
| PManagementController.saveChange | 语法 | public void saveChange(StaffVO vo); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被修改结束 |
| 后置条件 | 调用PManagement领域对象的saveChange方法 |
| PManagementController.save | 语法 | public void save(StaffVO vo); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被添加 |
| 后置条件 | 调用PManagement领域对象的save方法 |
| PManagementController.add | 语法 | public void add(); |
| 前置条件 | 在人员管理的过程中，添加按钮被点击 |
| 后置条件 | 调用PManagement领域对象的add方法 |
| PManagementController.getAllStaff | 语法 | public ArrayList<StaffVO> getAllStaff(); |
| 前置条件 | 开启新的一轮人员管理 |
| 后置条件 | 调用PManagement领域对象的getAllStaff方法 |
| PManagementController.endPManagement | 语法 | public void endPManagement(); |
| 前置条件 | 人员管理已开始 |
| 后置条件 | 调用PManagement领域对象的endPManagement方法 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| PManagement.select | 返回人员详细信息 |
| PManagement.delect(String id) | 根据id删除人员 |
| PManagement.delect(StaffVO vo) | 根据对象删除人员 |
| PManagement.revise(String id) | 根据ID返回可修改的人员信息单 |
| PManagement.saceChange | 保存修改后的人员信息单 |
| PManagement.save | 保存新增的人员信息单 |
| PManagement.add | 返回新的人员信息单 |
| PManagement.getAllStaff | 返回所有的人员信息单 |
| PManagement.endPManagement | 结束人员管理 |

Pmanagement的接口规范

|  |  |  |
| --- | --- | --- |
| PManagement.select | 语法 | public StaffVO select(String id); |
| 前置条件 | 开启新的一轮人员管理 |
| 后置条件 | 返回该人员的详细信息 |
| PManagement.delect | 语法 | public void delect(String id); |
| 前置条件 | 一开启新一轮的人员信息管理，已选择好需要删除的人员信息单，删除按钮被点击 |
| 后置条件 | 在一个人员管理回合中，删除该人员信息单，持久化更新本次删除的信息单，结束人员信息删除 |
| PManagement.revise | 语法 | public StaffVO revise(String id); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启人员信息单对象的修改方法 |
| PManagement.saveChange | 语法 | public void saveChange(StaffVO vo); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次人员管理的回合中，持久化更新人员信息单对象的数据，结束本次人员信息修改 |
| PManagement.save | 语法 | public void save(StaffVO vo); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次人员管理的回合中，持久化更新人员信息单对象的数据，结束本次人员信息添加 |
| PManagement.add | 语法 | public void add(); |
| 前置条件 | 在人员管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的人员信息单 |
| PManagement.getAllStaff | 语法 | public ArrayList<StaffVO> getAllStaff(); |
| 前置条件 | 开启新的一轮人员管理 |
| 后置条件 | 返回所有的人员信息单 |
| PManagement.endPManagement | 语法 | public void endPManagement(); |
| 前置条件 | 人员管理已开始 |
| 后置条件 | 结束本次人员管理，持久化更新涉及的领域对象的数据 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| PManagementDataService.find(String id) | 根据id返回的单一持久化对象 |
| PManagementDataService.insert(StaffPO po) | 插入单一持久化对象 |
| PManagementDataService.delect(StaffPO po) | 根据对象删除单一持久化对象 |
| PManagementDataService.delect(String id) | 根据id删除单一持久化对象 |
| PManagementDataService.update(StaffPO po) | 更新单一持久化对象 |
| PManagementDataService.findAll(); | 返回所有的持久化对象 |

AManagementController和AManagement的接口规范如下表

AManagementController的接口规范

|  |  |  |
| --- | --- | --- |
| AManagementController.select | 语法 | public AgencyVO select(String id); |
| 前置条件 | 已开启新的一轮机构管理 |
| 后置条件 | 调用AManagement领域对象的select方法 |
| AManagementController.delect | 语法 | public void delect(String id); |
| 前置条件 | 在一轮的机构信息管理，已选择好需要删除的机构信息单 |
| 后置条件 | 调用AManagement领域对像的delect方法 |
| AManagementContorller.revise | 语法 | public AgencyVO revise(String id); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被选择 |
| 后置条件 | 调用AManagement领域对象的revise方法 |
| AManagementController.saveChange | 语法 | public void saveChange(AgencyVO vo); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被修改结束 |
| 后置条件 | 调用AManagement领域对象的saveChange方法 |
| AManagementController.save | 语法 | public void save(AgencyVO vo); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被添加 |
| 后置条件 | 调用AManagement领域对象的save方法 |
| AManagementController.add | 语法 | public void add(); |
| 前置条件 | 在机构管理的过程中，添加按钮被点击 |
| 后置条件 | 调用AManagement领域对象的add方法 |
| AManagementController.getAllAgency | 语法 | public ArrayList<AgencyVO> getAllAgency(); |
| 前置条件 | 开启新的一轮机构管理 |
| 后置条件 | 调用AManagement领域对象的getAllAgency方法 |
| AManagementController.endAManagement | 语法 | public void endAManagement(); |
| 前置条件 | 机构管理已开始 |
| 后置条件 | 调用AManagement领域对象的endAManagement方法 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| AManagement.select | 返回机构详细信息 |
| AManagement.delect(String id) | 根据id删除机构 |
| AManagement.delect(AgencyVO vo) | 根据对象删除机构 |
| AManagement.revise(String id) | 根据ID返回可修改的机构信息单 |
| AManagement.saceChange | 保存修改后的机构信息单 |
| AManagement.save | 保存新增的机构信息单 |
| AManagement.add | 返回新的机构信息单 |
| AManagement.getAllAgency | 返回所有的机构信息单 |
| AManagement.endAManagement | 结束机构管理 |

AManagement的接口规范

|  |  |  |
| --- | --- | --- |
| AManagement.select | 语法 | public AgencyVO select(String id); |
| 前置条件 | 开启新的一轮机构管理 |
| 后置条件 | 返回该机构的详细信息 |
| AManagement.delect | 语法 | public void delect(String id); |
| 前置条件 | 已开启新一轮的机构信息管理，已选择好需要删除的机构信息单，删除按钮被点击 |
| 后置条件 | 在一个机构管理回合中，删除该机构信息单，持久化更新本次删除的信息单，结束机构信息删除 |
| AManagement.revise | 语法 | public AgencyVO revise(String id); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启机构信息单对象的修改方法 |
| AManagement.saveChange | 语法 | public void saveChange(AgencyVO vo); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次机构管理的回合中，持久化更新机构信息单对象的数据，结束本次机构信息修改 |
| AManagement.save | 语法 | public void save(AgencyVO vo); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次机构管理的回合中，持久化更新机构信息单对象的数据，结束本次机构信息添加 |
| AManagement.add | 语法 | public void add(); |
| 前置条件 | 在机构管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的机构信息单 |
| AManagement.getAllAgency | 语法 | public ArrayList<AgencyVO> getAllAgency(); |
| 前置条件 | 开启新的一轮机构管理 |
| 后置条件 | 返回所有的机构信息单 |
| AManagement.endAManagement | 语法 | public void endAManagement(); |
| 前置条件 | 机构管理已开始 |
| 后置条件 | 结束本次机构管理，持久化更新涉及的领域对象的数据 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| AManagementDataService.find(String id) | 根据id返回的单一持久化对象 |
| AManagementDataService.insert(AgencyPO po) | 插入单一持久化对象 |
| AManagementDataService.delect(AgencyPO po) | 根据对象删除单一持久化对象 |
| AManagementDataService.delect(String id) | 根据id删除单一持久化对象 |
| AManagementDataService.update(AgencyPO po) | 更新单一持久化对象 |
| AManagementDataService.findAll(); | 返回所有的持久化对象 |

DManagementController和DManagement的接口规范如下表

DManagementController的接口规范

|  |  |  |
| --- | --- | --- |
| DManagementController.select | 语法 | public DriverVO select(String id); |
| 前置条件 | 已开启新的一轮司机管理 |
| 后置条件 | 调用DManagement领域对象的select方法 |
| DManagementController.delect | 语法 | public void delect(String id); |
| 前置条件 | 在一轮的司机信息管理，已选择好需要删除的司机信息单 |
| 后置条件 | 调用DManagement领域对象的delect方法 |
| DManagementContorller.revise | 语法 | public DriverVO revise(String id); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被选择 |
| 后置条件 | 调用DManagement领域对象的revise方法 |
| DManagementController.saveChange | 语法 | public void saveChange(DriverVO vo); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被修改结束 |
| 后置条件 | 调用DManagement领域对象的saveChange方法 |
| DManagementController.save | 语法 | public void save(DiverVO vo); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被添加 |
| 后置条件 | 调用DManagement领域对象的save方法 |
| DManagementController.add | 语法 | public void add(); |
| 前置条件 | 在司机管理的过程中，添加按钮被点击 |
| 后置条件 | 调用DManagement领域对象的add方法 |
| DManagementController.getAllDriver | 语法 | public ArrayList<DriverVO> getAllDriver(); |
| 前置条件 | 开启新的一轮司机管理 |
| 后置条件 | 调用DManagement领域对象的getAllDriver方法 |
| DManagementController.endDManagement | 语法 | public void endDManagement(); |
| 前置条件 | 司机管理已开始 |
| 后置条件 | 调用DManagement领域对象的endDManagement方法 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| DManagement.select | 返回司机详细信息 |
| DManagement.delect(String id) | 根据id删除司机 |
| DManagement.delect(DriverVO vo) | 根据对象删除司机 |
| DManagement.revise(String id) | 根据ID返回可修改的司机信息单 |
| DManagement.saceChange | 保存修改后的司机信息单 |
| DManagement.save | 保存新增的司机信息单 |
| DManagement.add | 返回新的司机信息单 |
| DManagement.getAllDriver | 返回所有的司机信息单 |
| DManagement.endDManagement | 结束司机管理 |

DManagement的接口规范

|  |  |  |
| --- | --- | --- |
| DManagement.select | 语法 | public DriverVO select(String id); |
| 前置条件 | 开启新的一轮司机管理 |
| 后置条件 | 返回该司机的详细信息 |
| DManagement.delect | 语法 | public void delect(String id); |
| 前置条件 | 已开启新一轮的司机信息管理，已选择好需要删除的司机信息单，删除按钮被点击 |
| 后置条件 | 在一个司机管理回合中，删除该司机信息单，持久化更新本次删除的信息单，结束司机信息删除 |
| DManagement.revise | 语法 | public DriverVO revise(String id); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启司机信息单对象的修改方法 |
| DManagement.saveChange | 语法 | public void saveChange(DriverVO vo); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次司机管理的回合中，持久化更新司机信息单对象的数据，结束本次机构信息修改 |
| DManagement.save | 语法 | public void save(DriverVO vo); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次司机管理的回合中，持久化更新司机信息单对象的数据，结束本次机构信息添加 |
| DManagement.add | 语法 | public void add(); |
| 前置条件 | 在司机管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的机构信息单 |
| DManagement.getAllDriver | 语法 | public ArrayList<DriverVO> getAllDriver(); |
| 前置条件 | 开启新的一轮司机管理 |
| 后置条件 | 返回所有的司机信息单 |
| DManagement.endDManagement | 语法 | public void endDManagement(); |
| 前置条件 | 司机管理已开始 |
| 后置条件 | 结束本次司机管理，持久化更新涉及的领域对象的数据 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DManagementDataService.find(String id) | 根据id返回的单一持久化对象 |
| DManagementDataService.insert(DriverPO po) | 插入单一持久化对象 |
| DManagementDataService.delect(DriverPO po) | 根据对象删除单一持久化对象 |
| DManagementDataService.delect(String id) | 根据id删除单一持久化对象 |
| DManagementDataService.update(DriverPO po) | 更新单一持久化对象 |
| DManagementDataService.findAll(); | 返回所有的持久化对象 |

CManagementController和CManagement的接口规范如下表

CManagementController的接口规范

|  |  |  |
| --- | --- | --- |
| CManagementController.select | 语法 | public CarVO select(String id); |
| 前置条件 | 已开启新的一轮车辆管理 |
| 后置条件 | 调用CManagement领域对象的select方法 |
| CManagementController.delect | 语法 | public void delect(String id); |
| 前置条件 | 在一轮的车辆信息管理，已选择好需要删除的车辆信息单 |
| 后置条件 | 调用CManagement领域对像的delect方法 |
| CManagementContorller.revise | 语法 | public CarVO revise(String id); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被选择 |
| 后置条件 | 调用CManagement领域对象的revise方法 |
| CManagementController.saveChange | 语法 | public void saveChange(CarVO vo); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被修改结束 |
| 后置条件 | 调用CManagement领域对象的saveChange方法 |
| CManagementController.save | 语法 | public void save(CarVO vo); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被添加 |
| 后置条件 | 调用CManagement领域对象的save方法 |
| CManagementController.add | 语法 | public void add(); |
| 前置条件 | 在车辆管理的过程中，添加按钮被点击 |
| 后置条件 | 调用CManagement领域对象的add方法 |
| CManagementController.getAllCar | 语法 | public ArrayList<CarVO> getAllCar(); |
| 前置条件 | 开启新的一轮车辆管理 |
| 后置条件 | 调用CManagement领域对象的getAllCar方法 |
| CManagementController.endCManagement | 语法 | public void endCManagement(); |
| 前置条件 | 车辆管理已开始 |
| 后置条件 | 调用CManagement领域对象的endCManagement方法 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| CManagement.select | 返回车辆详细信息 |
| CManagement.delect(String id) | 根据id删除车辆 |
| CManagement.delect(CarVO vo) | 根据对象删除车辆 |
| CManagement.revise(String id) | 根据ID返回可修改的车辆信息单 |
| CManagement.saceChange | 保存修改后的车辆信息单 |
| CManagement.save | 保存新增的车辆信息单 |
| CManagement.add | 返回新的车辆信息单 |
| CManagement.getAllCar | 返回所有的车辆信息单 |
| CManagement.endCManagement | 结束车辆管理 |

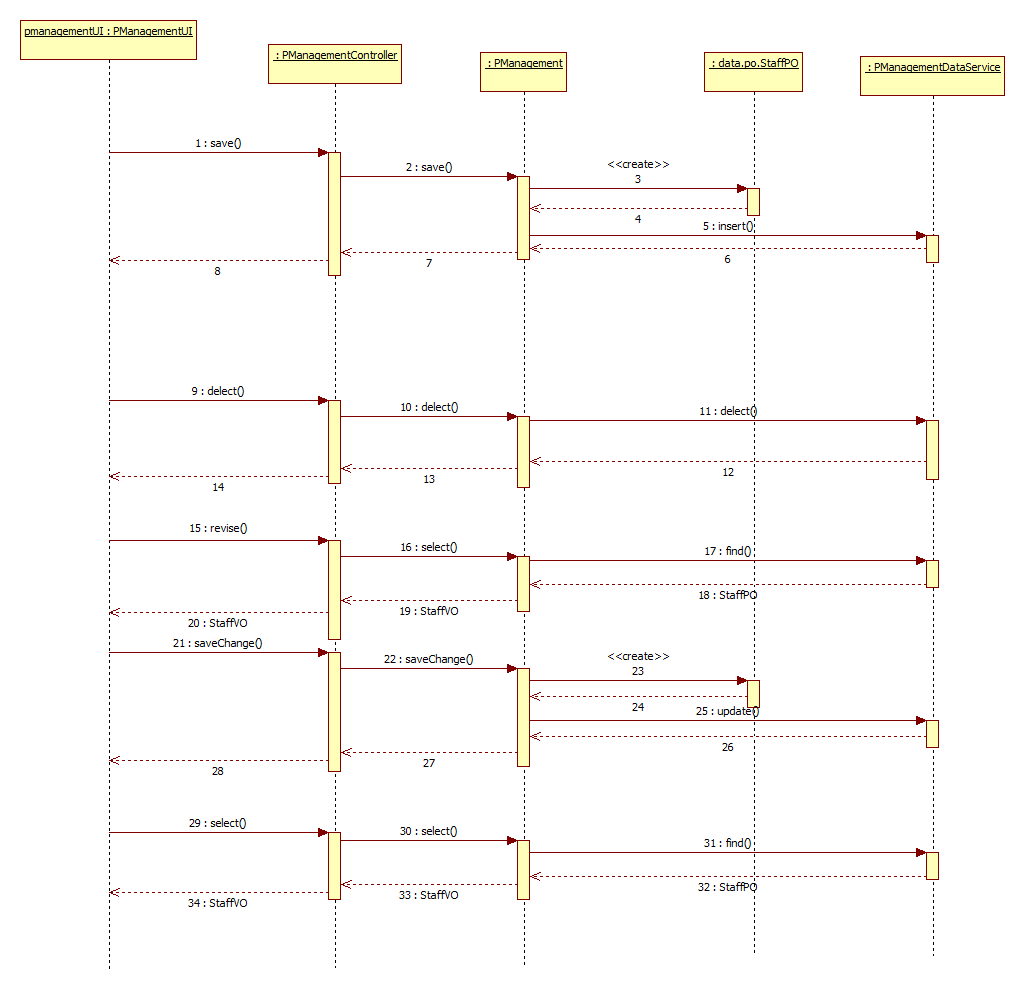
CManagement的接口规范

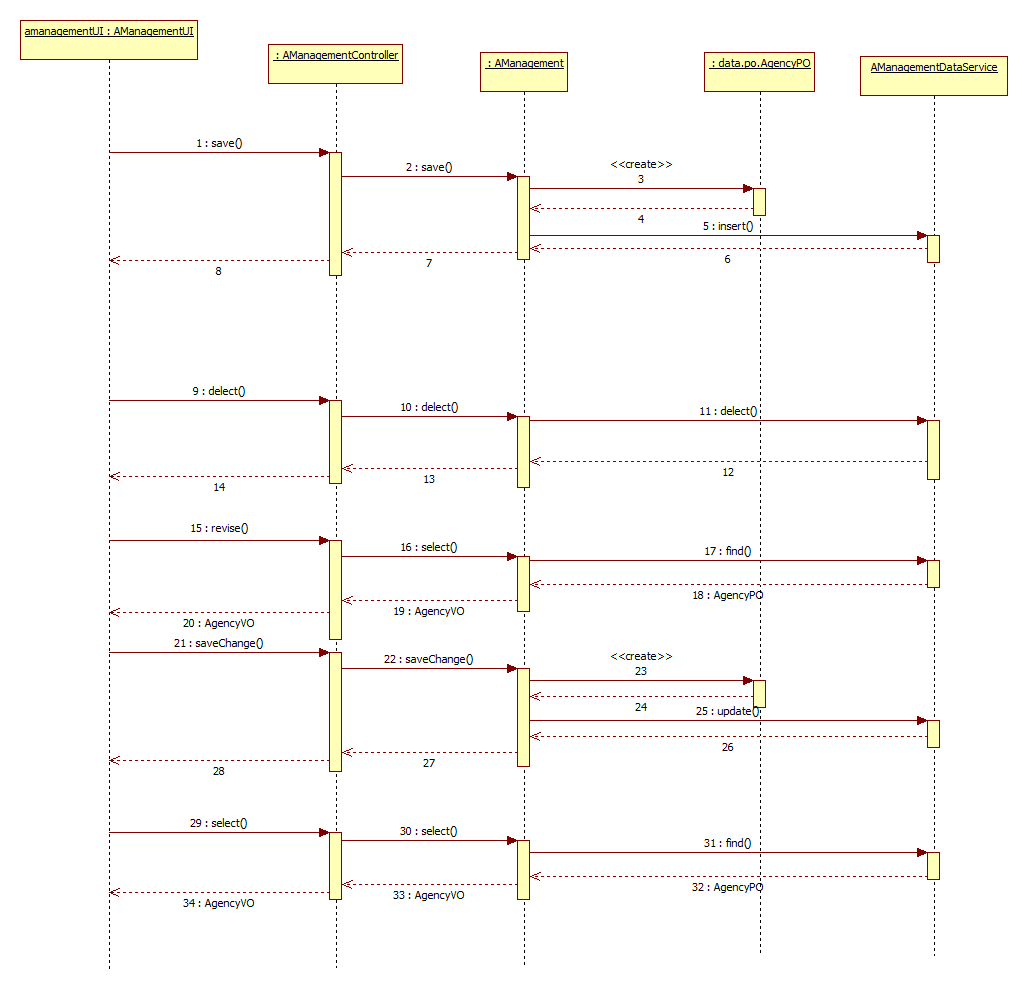
|  |  |  |
| --- | --- | --- |
| CManagement.select | 语法 | public CarVO select(String id); |
| 前置条件 | 开启新的一轮车辆管理 |
| 后置条件 | 返回该车辆的详细信息 |
| CManagement.delect | 语法 | public void delect(String id); |
| 前置条件 | 已开启新一轮的车辆信息管理，已选择好需要删除的车辆信息单，删除按钮被点击 |
| 后置条件 | 在一个车辆管理回合中，删除该车辆信息单，持久化更新本次删除的信息单，结束车辆信息删除 |
| CManagement.revise | 语法 | public CarVO revise(String id); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启车辆信息单对象的修改方法 |
| CManagement.saveChange | 语法 | public void saveChange(CarVO vo); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次车辆管理的回合中，持久化更新车辆信息单对象的数据，结束本次车辆信息修改 |
| CManagement.save | 语法 | public void save(CarVO vo); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次车辆管理的回合中，持久化更新车辆信息单对象的数据，结束本次车辆信息添加 |
| CManagement.add | 语法 | public void add(); |
| 前置条件 | 在车辆管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的车辆信息单 |
| CManagement.getAllCar | 语法 | public ArrayList<CarVO> getAllCar(); |
| 前置条件 | 开启新的一轮车辆管理 |
| 后置条件 | 返回所有的车辆信息单 |
| CManagement.endCManagement | 语法 | public void endCManagement(); |
| 前置条件 | 车辆管理已开始 |
| 后置条件 | 结束本次车辆管理，持久化更新涉及的领域对象的数据 |

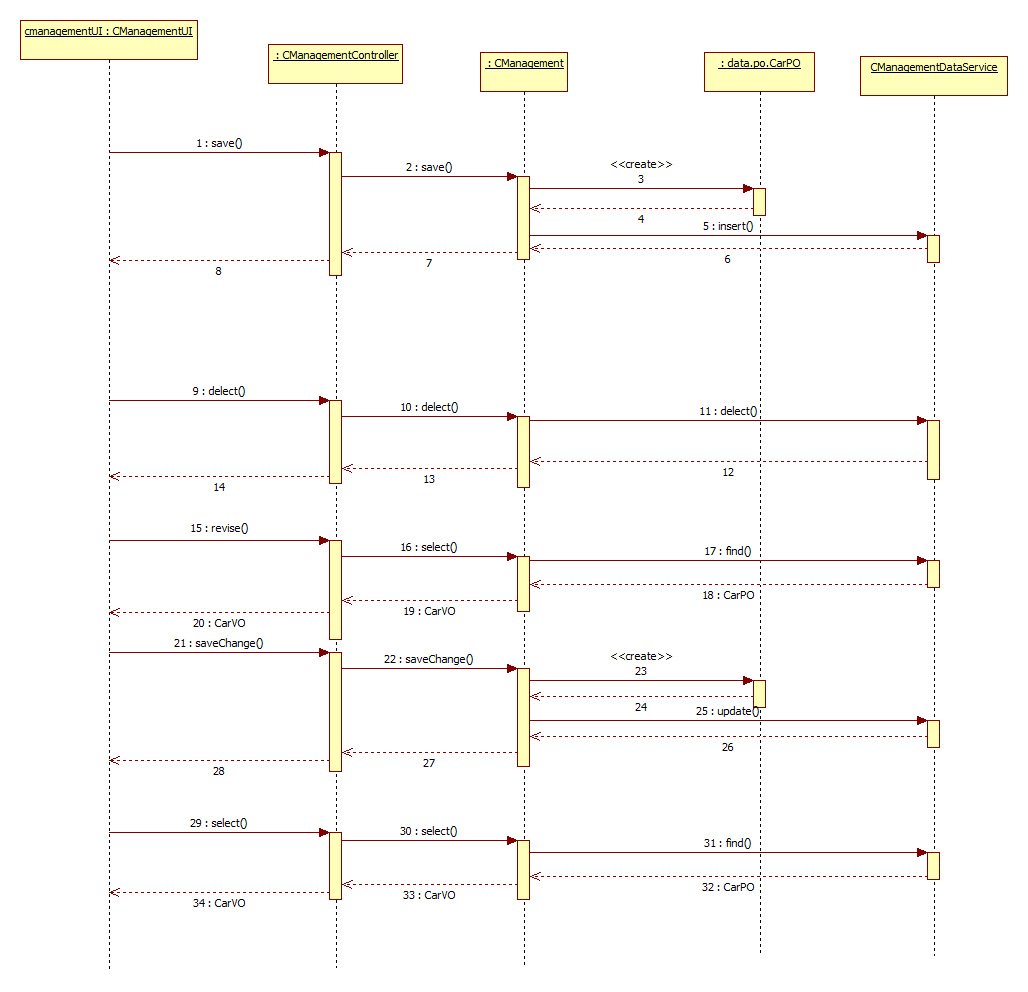
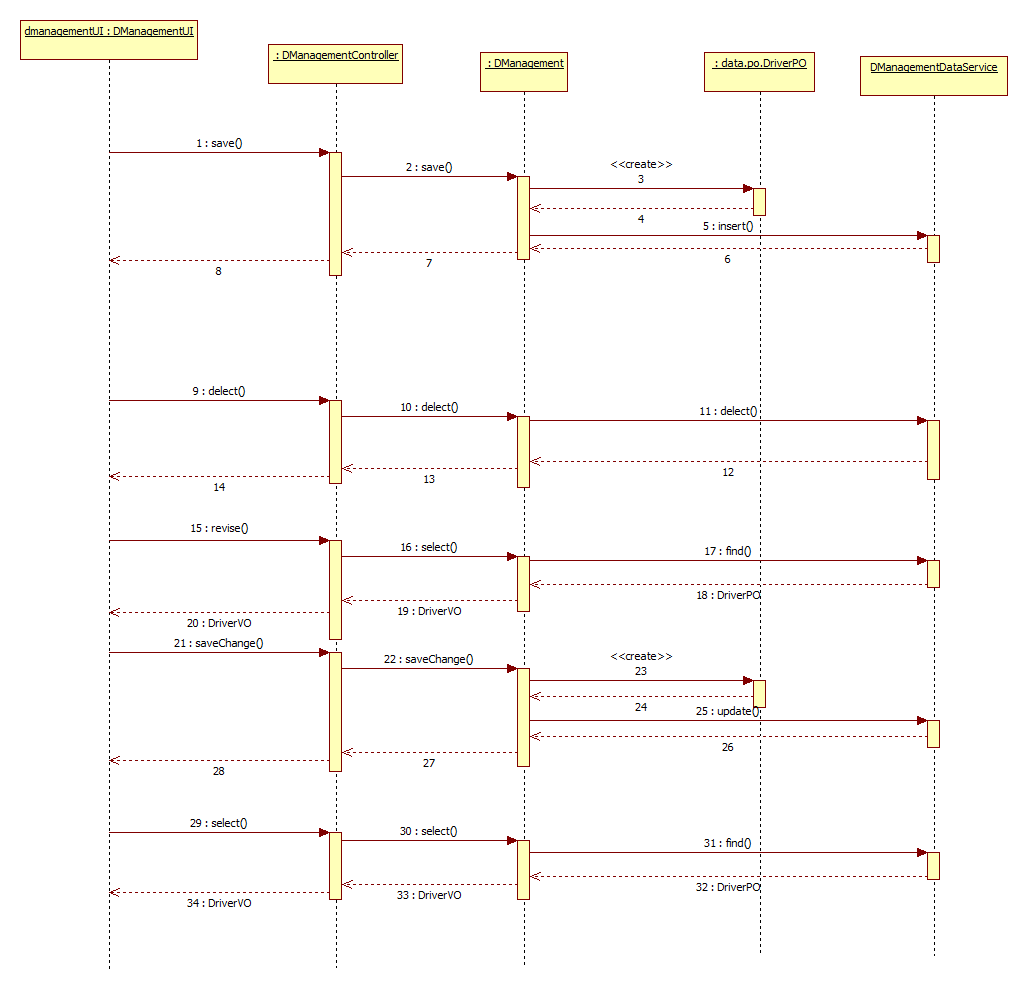
需要的服务（需接口）

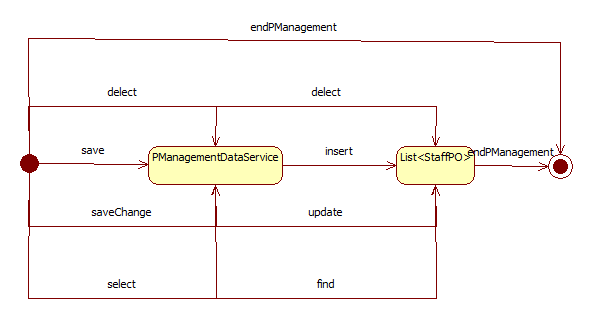
|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| CManagementDataService.find(String id) | 根据id返回的单一持久化对象 |
| CManagementDataService.insert(CarPO po) | 插入单一持久化对象 |
| CManagementDataService.delect(CarPO po) | 根据对象删除单一持久化对象 |
| CManagementDataService.delect(String id) | 根据id删除单一持久化对象 |
| CManagementDataService.update(CarPO po) | 更新单一持久化对象 |
| CManagementDataService.findAll(); | 返回所有的持久化对象 |

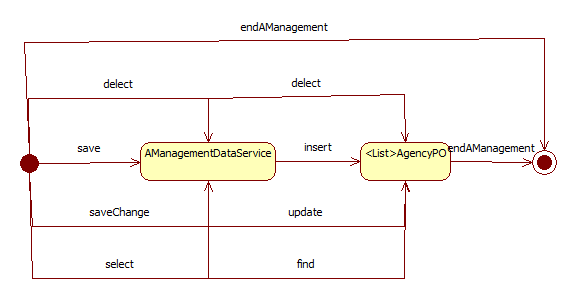
(4)pamanagementsl模块的动态模型

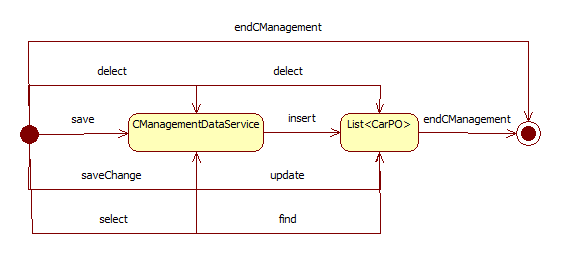
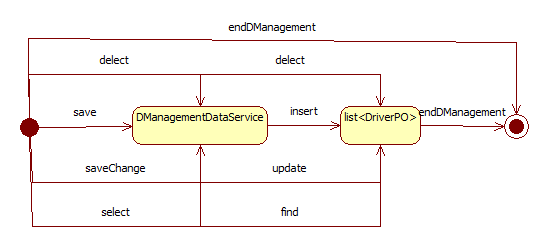












4.2用户界面层的行为

在界面层中界面的跳转的实现，我们决定采取委托式的控制风格，在快递物流洗头膏中业务逻辑层的设计可以分为两个子层：一个是Controller；一个是domain Object层。Controller层主要负责向界面层(presentation)提供服务，并且利用具体的damian Object来实现各个服务。Domain Object是业务逻辑层中各个领域模型。下图为pamanagementbl逻辑层基于委托式控制风格的业务逻辑设计

