

**软件学院大作业任务书**

课程名称： 统一建模语言UML

题 目： 多功能疲劳驾驶检测系统

专 业： 软件工程

班 级： 软工177

学 号： 8002117273 8002117256 8002117255 8002117277

学生姓名： 申海涛 吴波 龚殿城 宁志成

完成人数： 4 人

起讫日期： 2019-9-5

任课教师： 周翔 职称： 讲师

部分管主任： 徐健峰

完成时间： 2019-12-2

目录

[一、 概述 3](#_Toc26300427)

[1. 项目背景 3](#_Toc26300428)

[2. 编写目的 5](#_Toc26300429)

[3. 软件定义 6](#_Toc26300430)

[二、 需求分析 6](#_Toc26300431)

[1. 问题陈述 6](#_Toc26300432)

[2. 需求分析 7](#_Toc26300433)

[功能需求 8](#_Toc26300434)

[非功能需求 8](#_Toc26300435)

[三、 功能模型设计 10](#_Toc26300436)

[四、 动态模型设计 16](#_Toc26300437)

[1. 活动图 16](#_Toc26300438)

[2. 交互图 26](#_Toc26300439)

[3. 状态图 32](#_Toc26300440)

[五、 逻辑模型设计 36](#_Toc26300441)

[六、 物理模型设计 44](#_Toc26300442)

[1. 组件图 44](#_Toc26300443)

[2. 部署图 47](#_Toc26300444)

[3. 数据模型图 48](#_Toc26300445)

[4. 正向工程 51](#_Toc26300446)

[七、 总结 54](#_Toc26300447)

[八、 参考文献 57](#_Toc26300448)

# 概述

## 项目背景

伴随着我国经济的飞速发展，人民的生活水平不断提高，作为现在最为常见的交通工具，汽车早已来到了大多数人的面前。据公安部统计，2018年全国新注册登记机动车3172万辆，机动车保有量已达3.27亿辆，其中汽车2.4亿辆，小型载客汽车首次突破2亿辆；机动车驾驶员突破4亿人，达4.09亿人，其中汽车驾驶员3.69亿人。作为一种方便快捷的交通工具，汽车让我们的生活质量提高了很多，但同时也把我们置入安全隐患之中。交通事故四个字看似很遥远，但每天你都可以在电视或者其他新闻媒体上看到交通事故的报道，轻者伤筋动骨，重者家破人亡。

根据官方报告，2015年，我国共发生187781起交通事故，造成58022人死亡，199880人受伤，财产损失103692万元；2016年，我国汽车发生交通事故212846起，造成63093人死亡，226430人受伤，直接经济损失12.1亿元；2017年，我国汽车发生交通事故139412起，导致46817人死亡，139180人受伤，直接财产损失103978万元。每一个数据都代表的是一个家庭的痛苦，我们无法从这些数据上感受到他们的痛苦和艰难，但从数据的庞大上，我们应该感到震撼。这就是为什么我们需要多功能疲劳驾驶检测系统的原因。

## 编写目的

如今城市化生活水平大幅度提升，居民所拥有的汽车越来越多，因此，由于没有很好的办法处理疲劳驾驶问题，交通事故的发生越发的频繁。所以我们选择设计此系统的目的主要有两点：

1. 提高判断驾驶者是否疲劳驾驶的能力，通过判断是否疲劳驾驶，可以采取一系列措施降低因疲劳驾驶而产生的交通事故概率。
2. 降低发生交通事故的概率，通过语音提醒与导航，将疲劳驾驶者安全送达目的地休息，避免交通事故的发生。

# 需求分析

## 问题陈述

随着科技与经济的高速发展，机动车辆已成为现代交通的重要参与者。据公安部统计，2018年全国新注册登记机动车3172万辆，机动车保有量已达3.27亿辆，其中汽车2.4亿辆，小型载客汽车首次突破2亿辆；机动车驾驶员突破4亿人，达4.09亿人，其中汽车驾驶员3.69亿人。作为一种方便快捷的交通工具，汽车让我们的生活质量提高了很多，而交通事故却是造成人类伤亡的主要原因之一，其中疲劳驾驶造成的交通事故占总数的20%左右，占特大交通事故的40%以上。传统的车辆被动安全防护措施已经无法满足现代复杂的驾驶状况，驾驶员的驾驶行为本身是以视觉处理为基础的，所以通过基于机器视觉的辅助驾驶系统已成为解决当前严重的交通安全问题的重要途经之一。现有的疲劳驾驶监测技术大多数基于驾驶员的行为特征，单从驾驶员行为特征判断疲劳的稳定性较差，通过引入集成的行车记录仪，利用检测的道路信息，将两者信息融合做出疲劳驾驶判断，其稳定性和准确性会有所提高。疲劳驾驶监测技术已经较为成熟，产品也已经上线，而用户不买账的大部分原因是不想在空间已经较为局促的车上再多加一套设备。因此，将疲劳检测系统和行车记录集成为一体，在一定程度上可以降低交通事故的发生率，减少交通事故带来的危害，使驾驶员驾驶和公共出行更加安全。

## 需求分析

本系统从驾驶开始时便对驾驶员的操作行为进行记录，并能够通过识别长途旅行中驾驶操作的变化，对驾驶员的疲劳程度进行判断。驾驶员转向操作频率变低，并伴随轻微但急骤的转向动作以保持行驶方向，是驾驶精力不集中的典型表现。根据以上动作的出现频率，并综合诸如旅途长度、转向灯使用情况、驾驶时间等其他参数，系统对驾驶员的疲劳程度进行计算和鉴别，如果计算结果超过某一定值，智能机器人提示驾驶员需要休息，并根据实时路况，给出最近的休息点，引导驾驶员停车休息。同时，将提醒附近车辆，该车驾驶员疲劳，需提高警惕，保持车距。

# 功能模型设计

（一）需求

多功能疲劳驾驶检测系统的需求描述如下：

该系统参与者：  
驾驶员：与多功能驾驶安全监测系统直接关联，受益者  
系统管理员：管理员：管理员是本多功能驾驶安全检测系统的开发者和维护者，主要负责接收反馈、检查、维护、更新系统等功能，能够让本系统在第一时间进行改善，保证本系统功能的准确，及时检测和分析出各项数据并发出相应提示，确保司机安全驾驶。  
图灵机器人：提供语音服务，能够与驾驶员进行人机对话，分析路线情况，指挥驾驶员到指定地点停车休息。

（二）分析

本系统拟使用J2EE语言和SSM框架通过三层模型实现：数据核心层、业务逻辑层和表现层。其中，数据核心层包括对数据库的操作；业务逻辑层作为中间层对用户输入进行逻辑处理，再映射到相应的数据层操作；表现层包括用户界面。

本系统涉及到的用户包括：驾驶员与交警部门。由于这两类用户的基本信息相似，所以可以抽象出基用户。驾驶员与交警部门则从用户统一派生。

（三）事件流

该部分内容详见《多功能疲劳驾驶检测系统用例分析文档》。

（四）用例图

为了将更清晰地展示该系统的功能，我们将用例图分层进行逐步细化，简化问题，如图4-0所示。

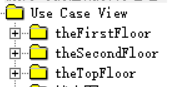


图4-0

4.1顶层用例图

4.1.1顶层用例如图4-1-1所示。



图4-1-1

4.1.2顶层用例图如图4-1-2所示。

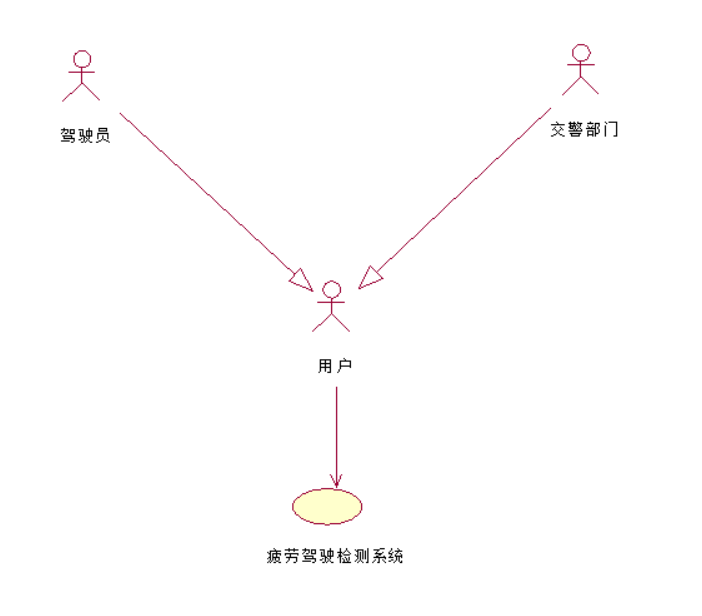


图4-1-2

4.2一层用例图

4.2.1一层用例如图4-2-1所示。

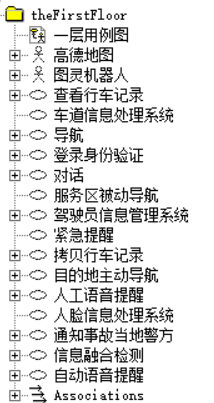


图4-2-1

4.2.2一层用例图如图4-2-2所示。

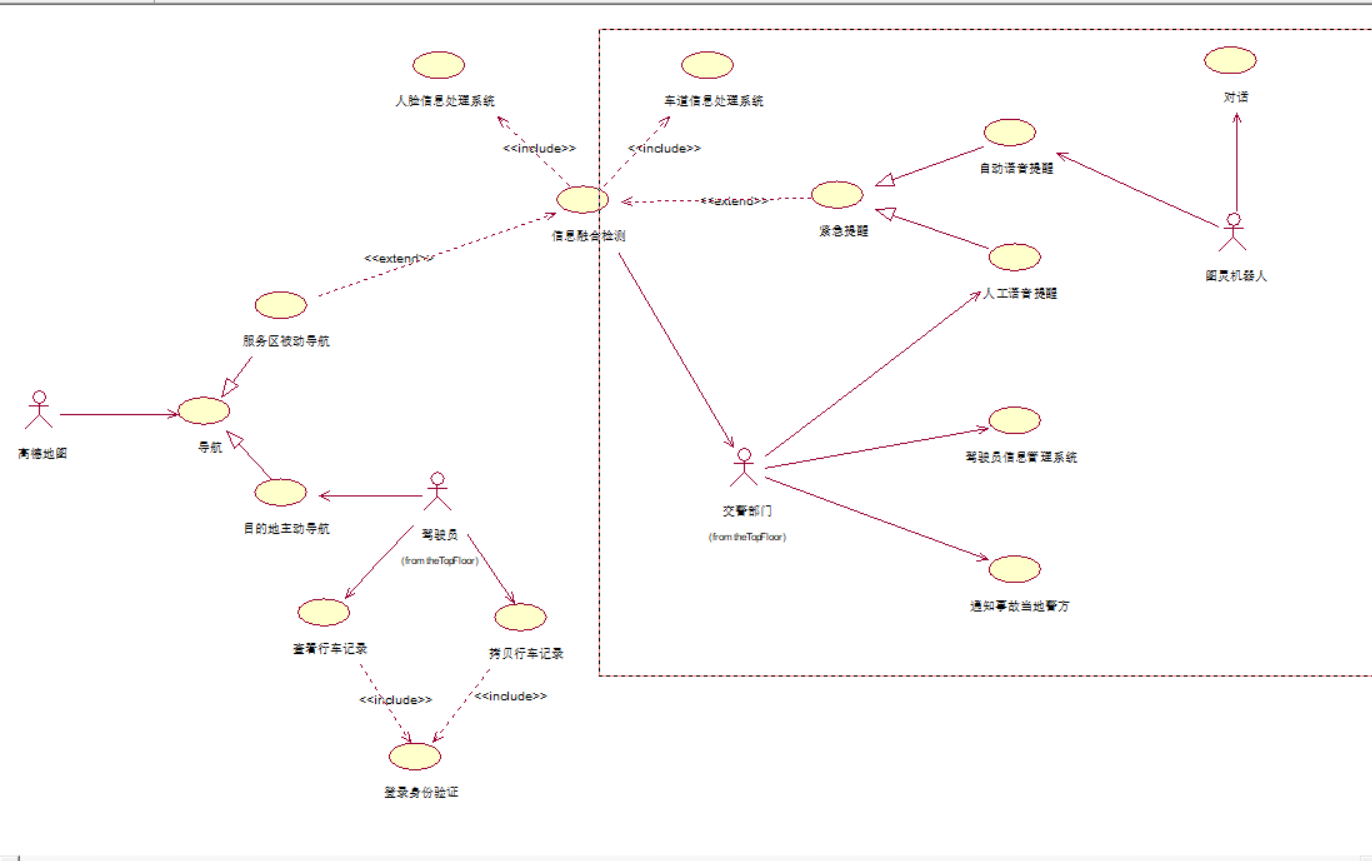


图4-2-2

4.3二层用例图

4.3.1二层用例如图4-3-1所示。

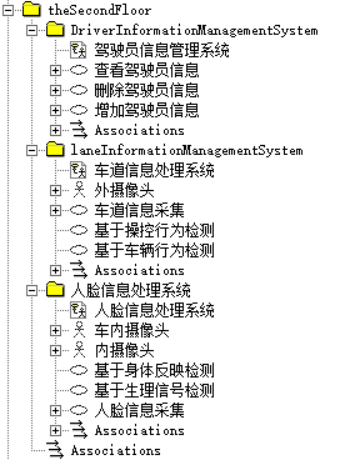


图4-3-1

4.3.2驾驶员信息管理系统二层用例图，如图4-3-2所示。

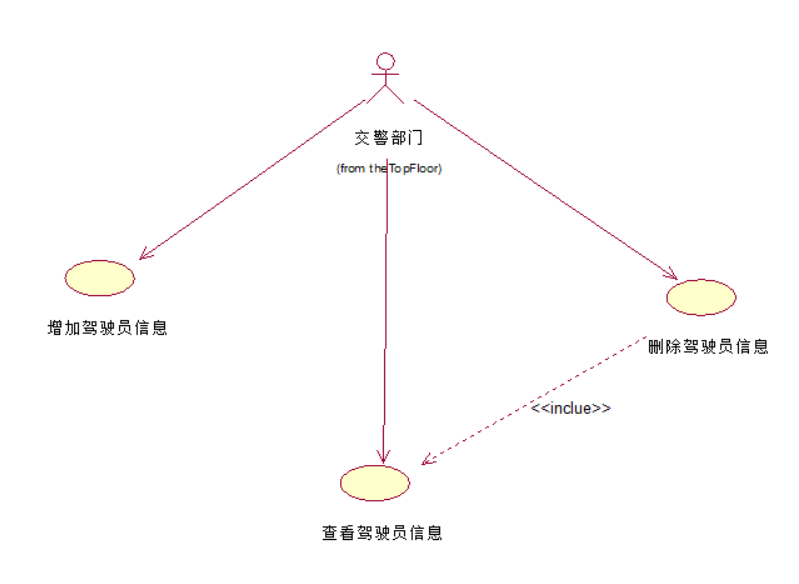


图4-3-2

4.3.3车道信息处理系统用例图，如图4-3-3所示。

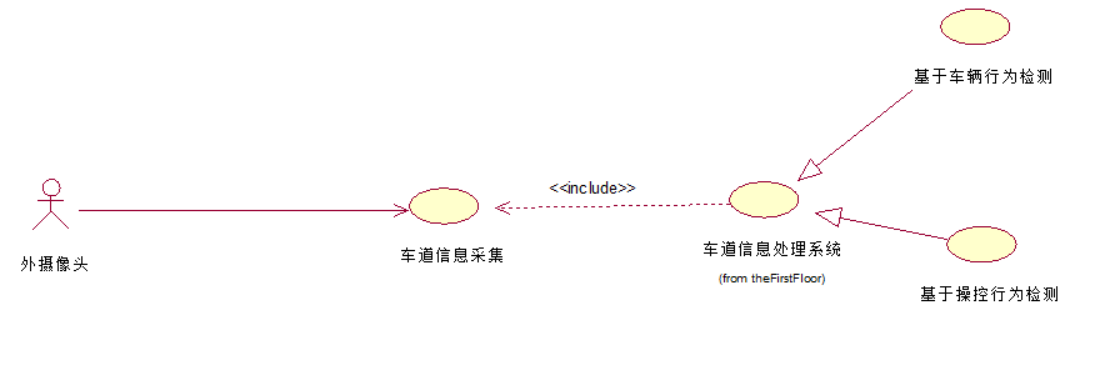


图4-3-3

4.3.人脸信息管理系统用例图，如图4-3-4所示。

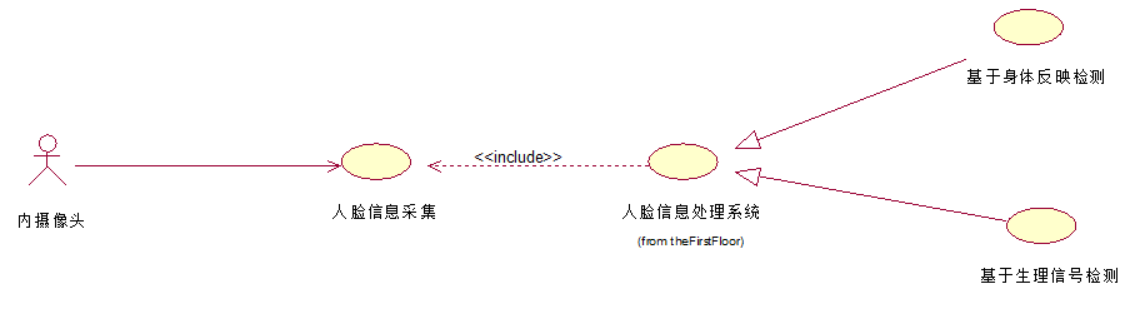


图4-3-4

# 动态模型设计

## 活动图

1. 需求分析

针对用例图中的一层用例以及二层用例，使用活动图对用例实现建模。

活动图强调的是从活动到活动的控制流。

使用活动图描述活动的顺序，展现从一个活动到另一个活动的控制流。

活动图着重表现从一个活动到另一个活动的控制流，充分直观的展示出系统的内部处理驱动的流程。

针对本系统一层用例使用活动图进行动态建模的的有：查看行车记录，导航，登陆身份验证，拷贝行车记录，人工语音提醒，通知事故当地警方，信息融合检测，自动语音提醒。

针对本系统二层用例使用活动图进行动态建模的有：增加/查看/删除驾驶员信息；人脸信息采集；车道信息采集。

1. 事件流

该部分内容详见《多功能疲劳驾驶检测系统用例分析文档》。

1. 活动图

3.1一层活动图

（1）1.1查看行车记录

如图1-1：

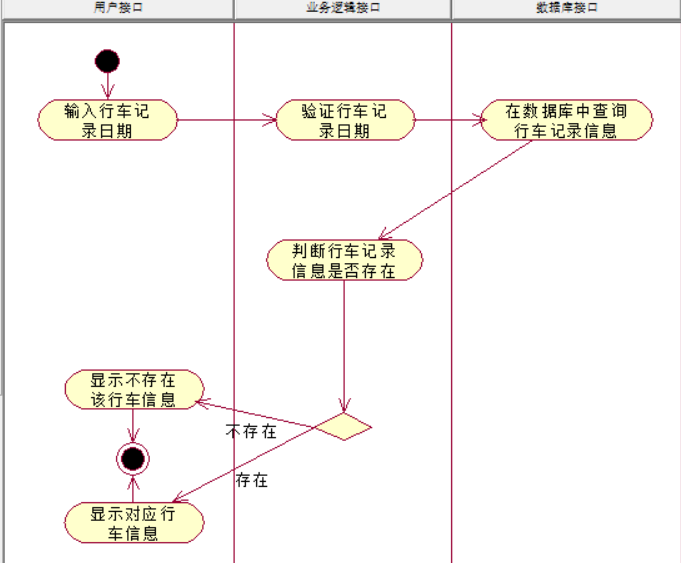


图1-1

（2）导航

如图1-2所示：

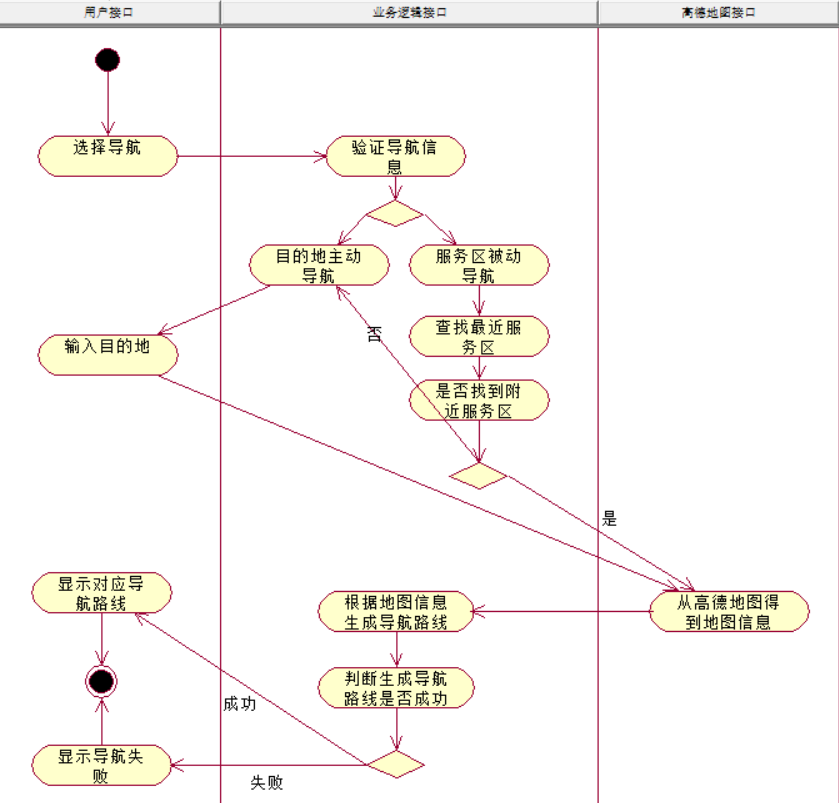
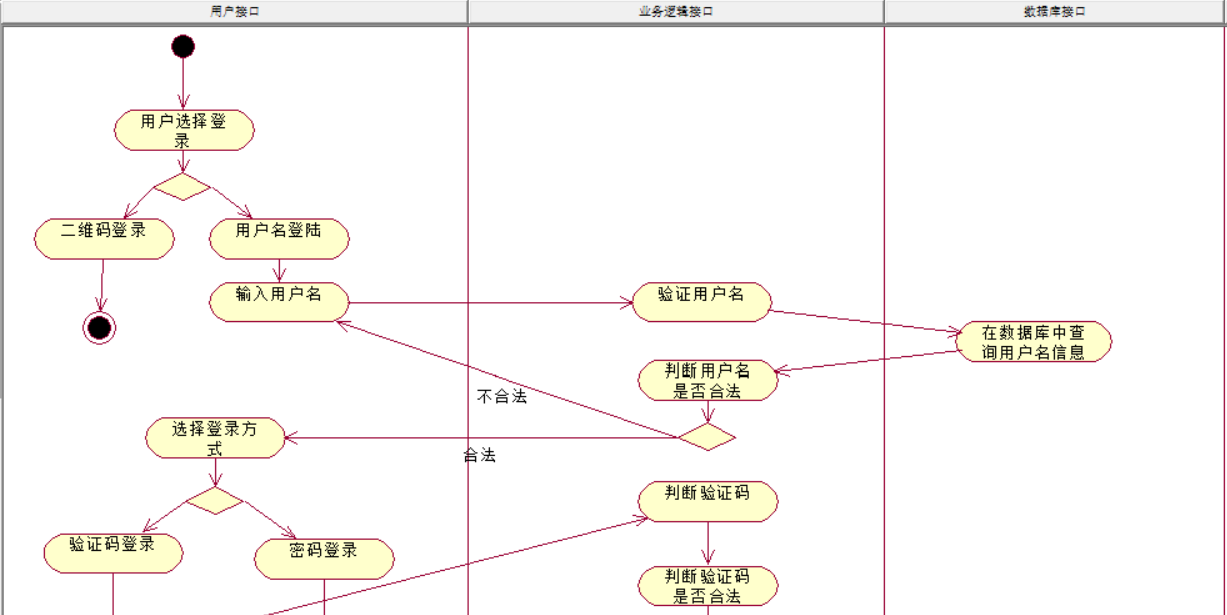


图1-2

（3）登陆身份验证

如图1-3所示：



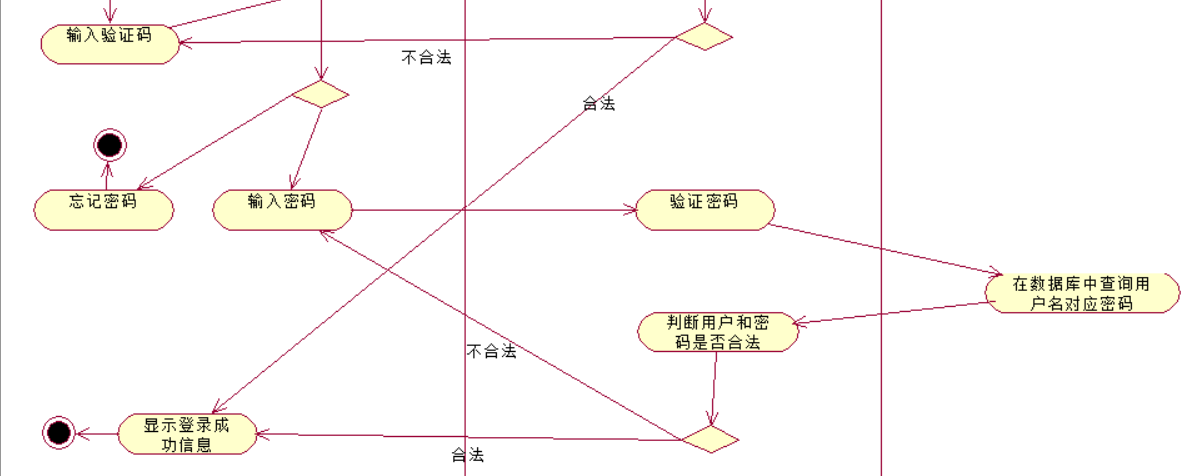


图1-3

（4）拷贝行车记录

如图1-4所示：

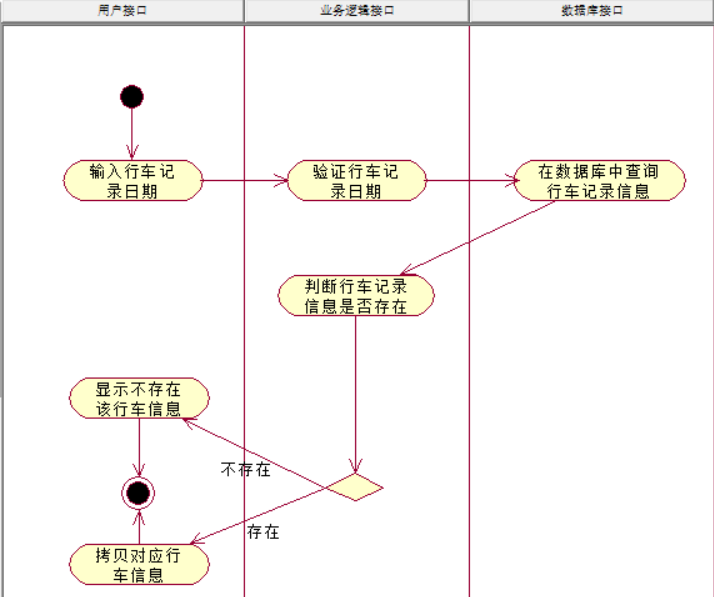


图1-4

（5）人工语音提醒

如图1-5所示：

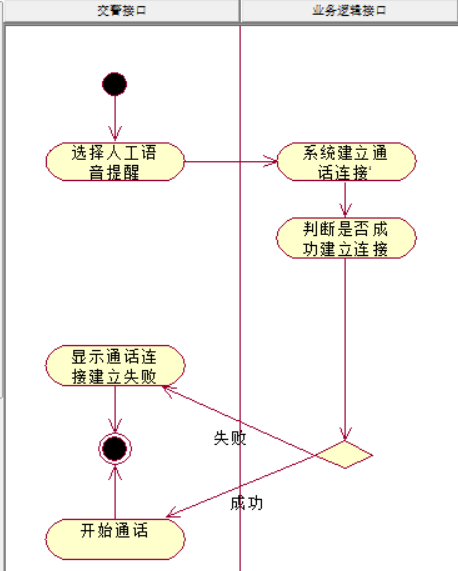
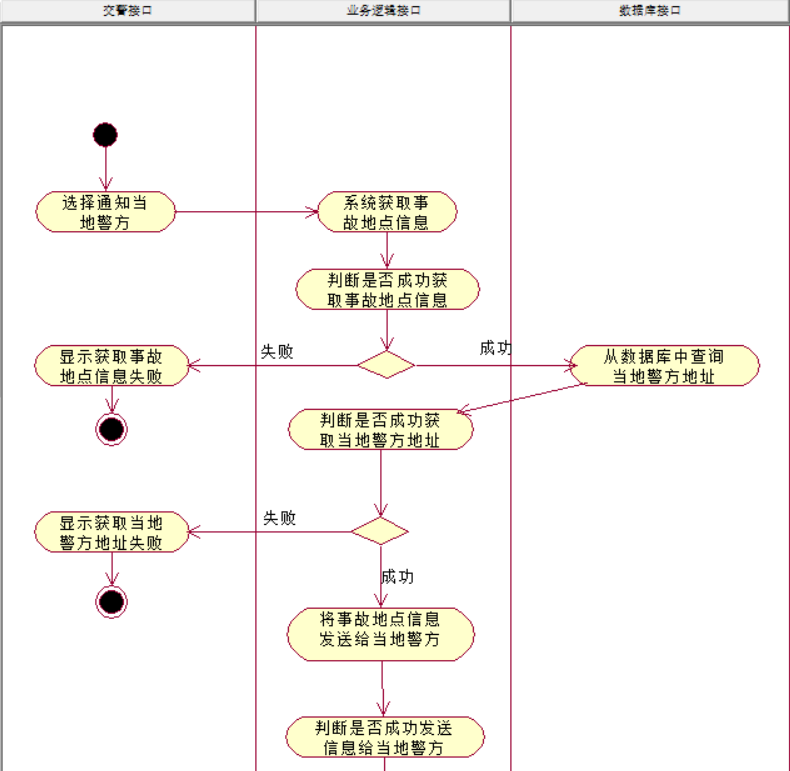


图1-5

（6）通知事故当地警方

如图1-6所示：



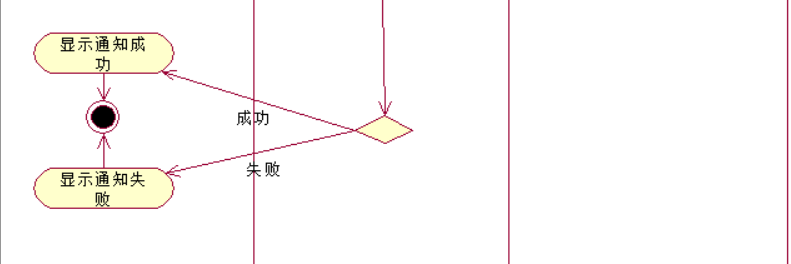
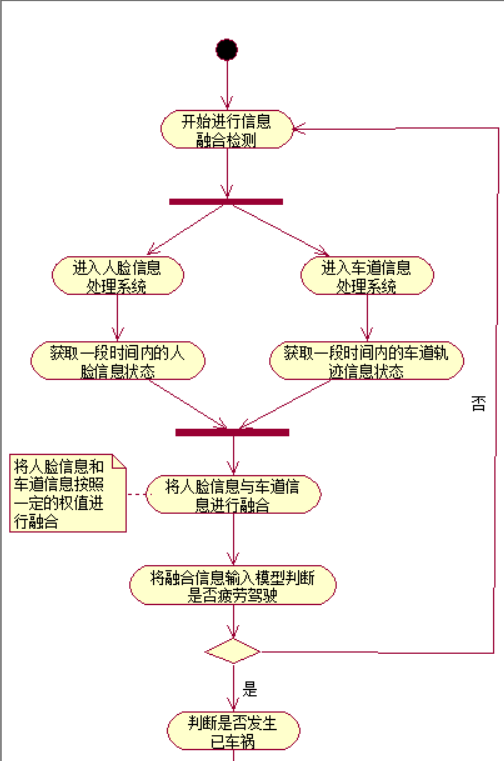


图1-6

（7）信息融合检测

如图1-7所示：



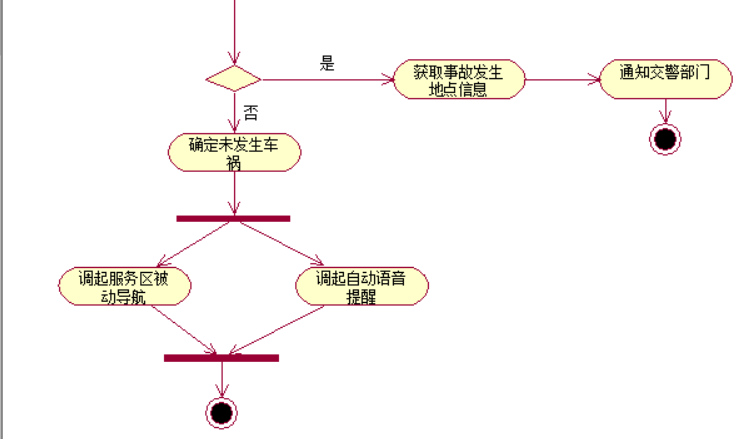


图1-7

（8）自动语音提醒

如图1-8所示：

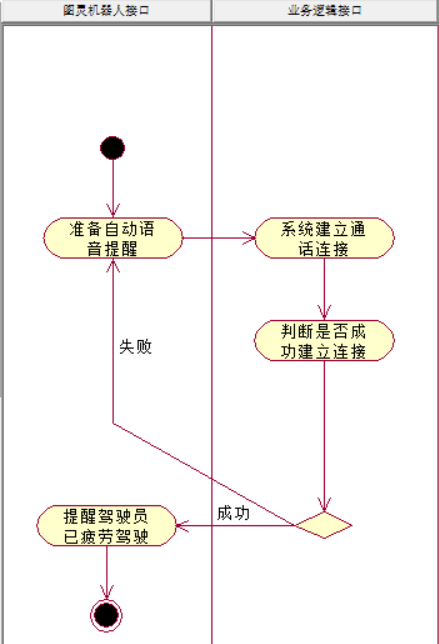


图1-8

3.2二层活动图

以下具体列举二层活动图：

（1）车道信息采集

如图2-1所示：

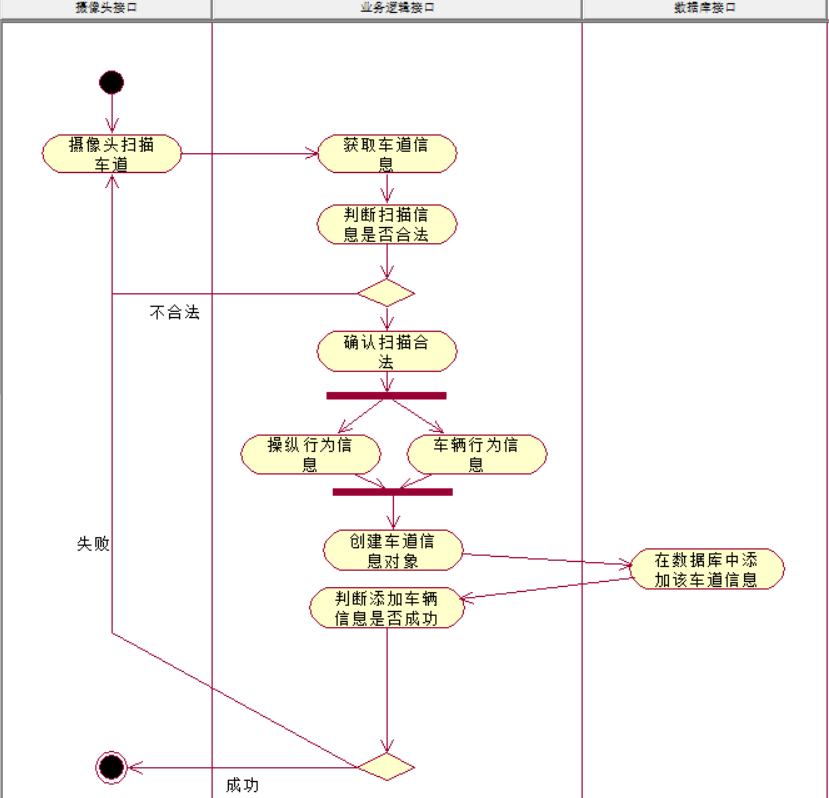


图2-1

（2）人脸信息采集

如图2-2所示：

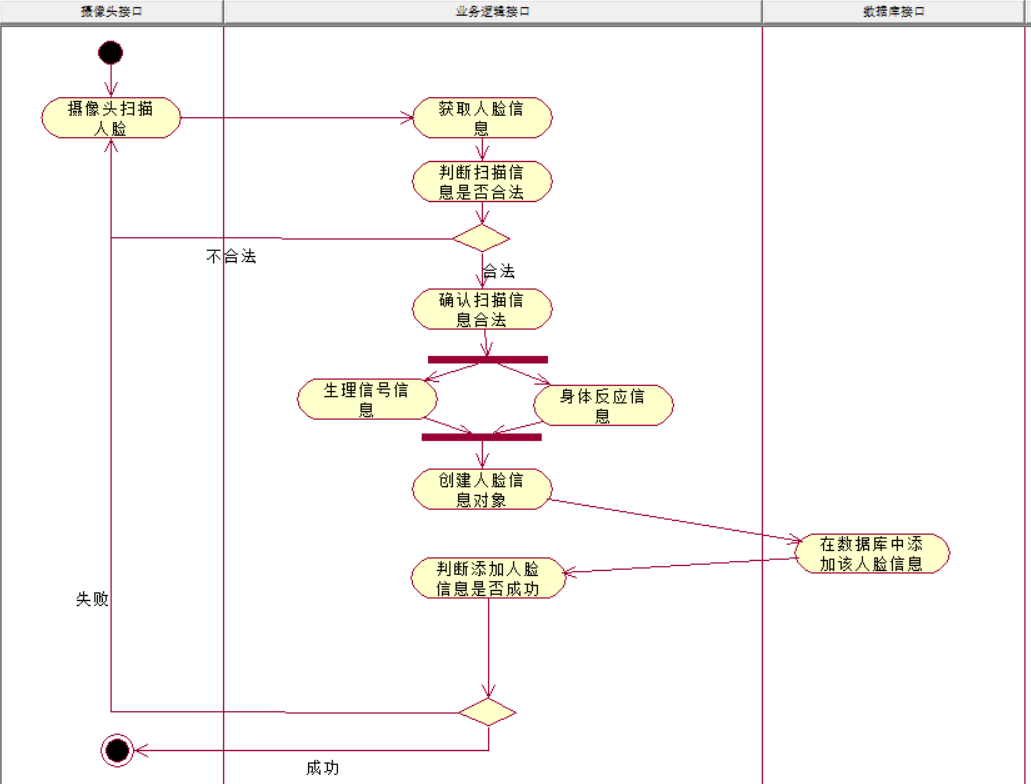


图2-2

（3）查看驾驶员信息

如图2-3所示：

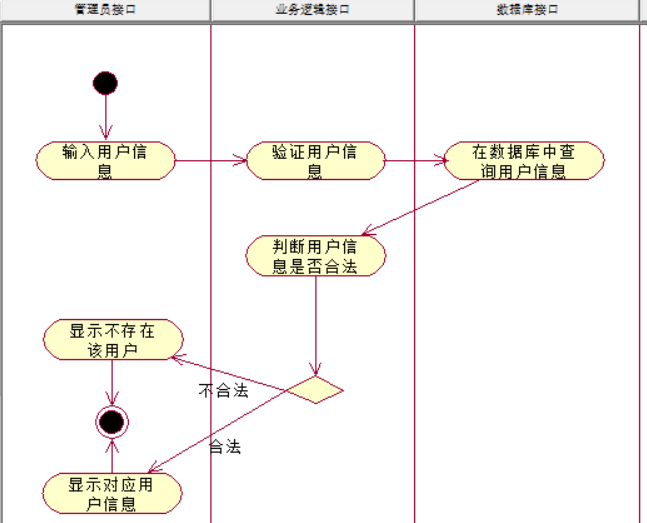


图2-3

（4）删除驾驶员

如图2-4所示：

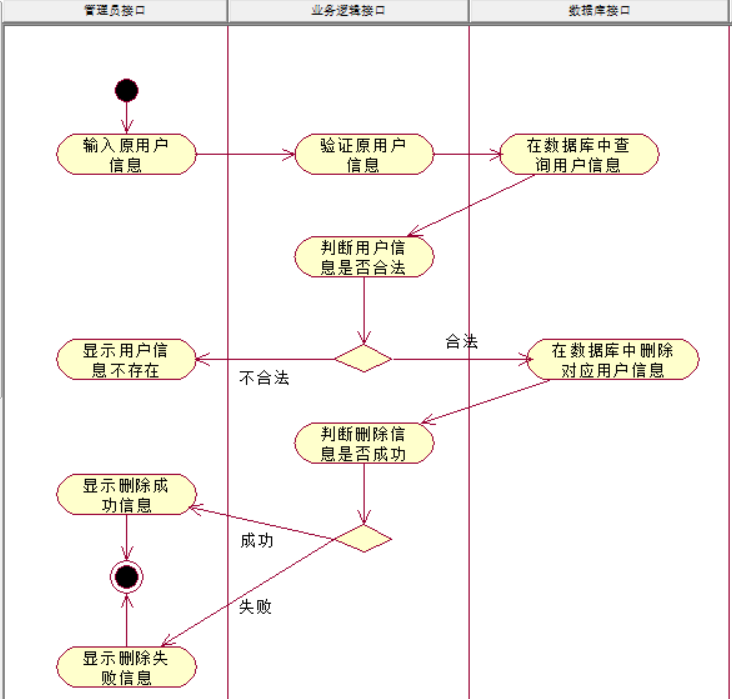


图2-4

（5）增加驾驶员

如图2-5所示：

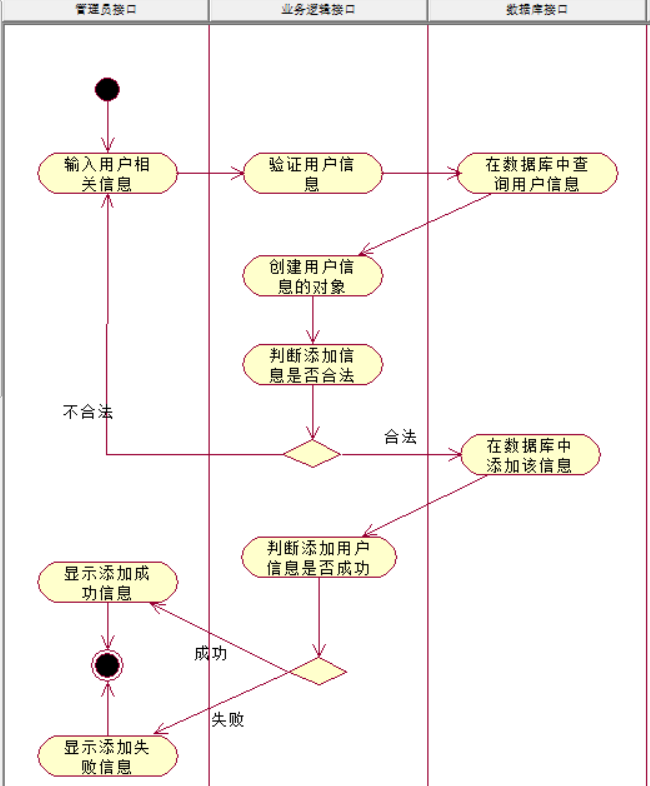
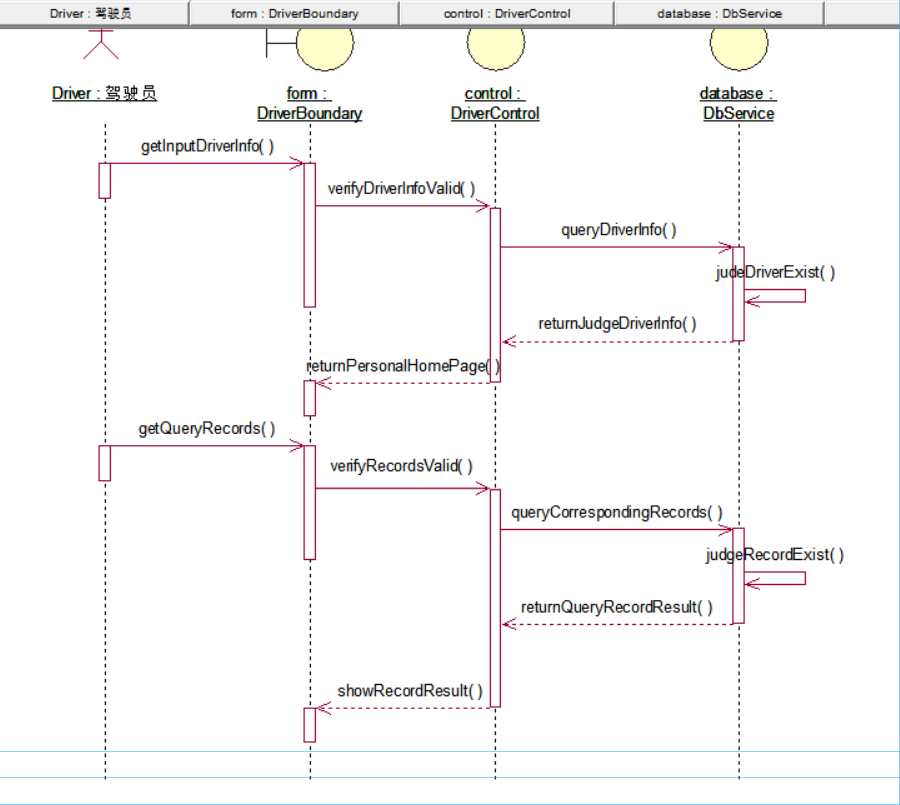
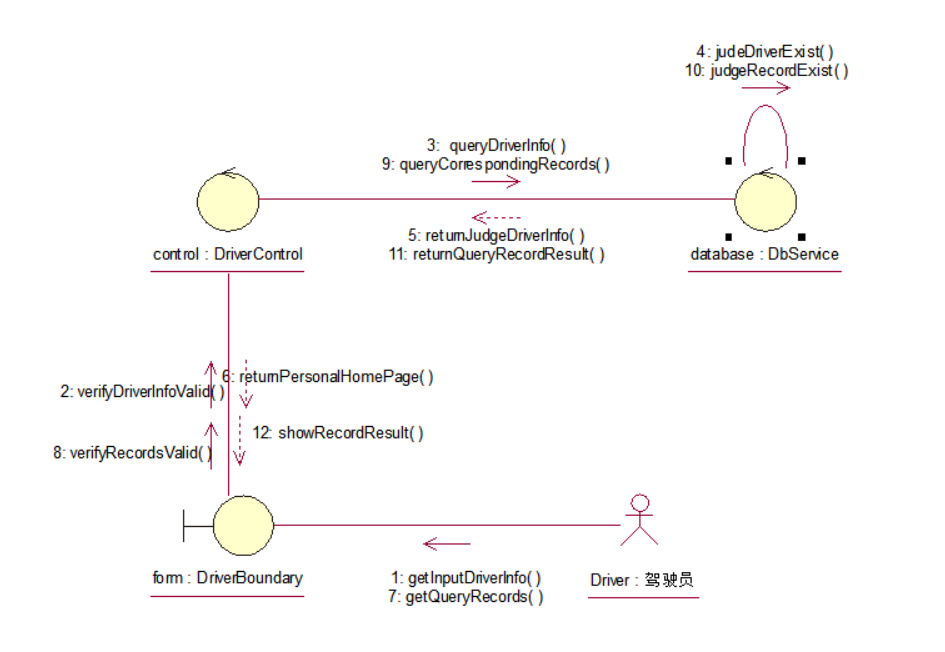


图2-5

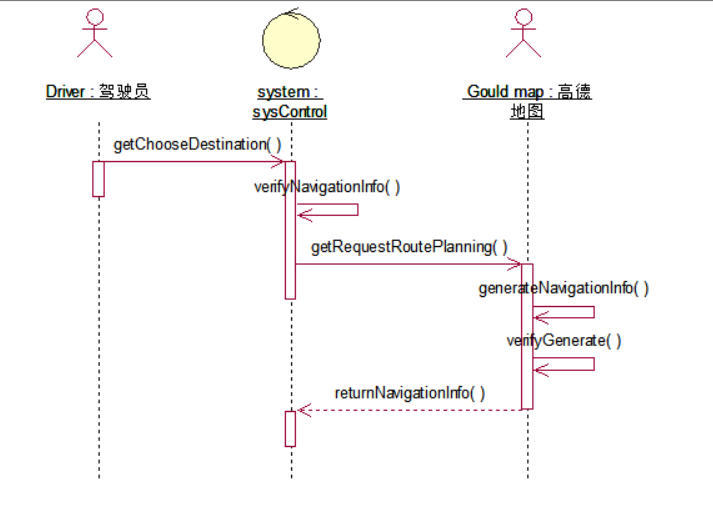
## 交互图

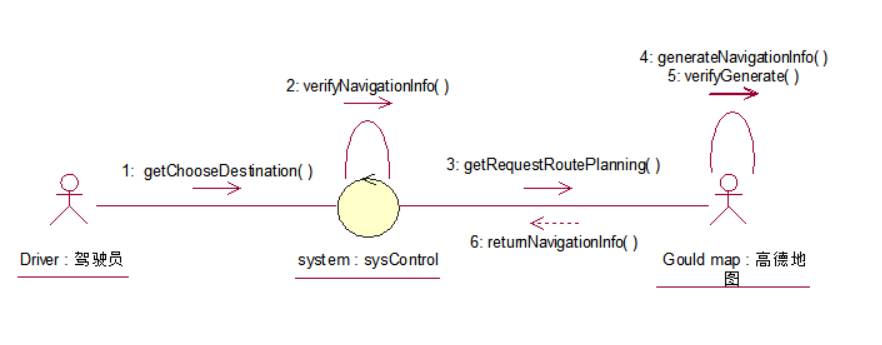
1.一层序列图及协作图——查看行车记录:



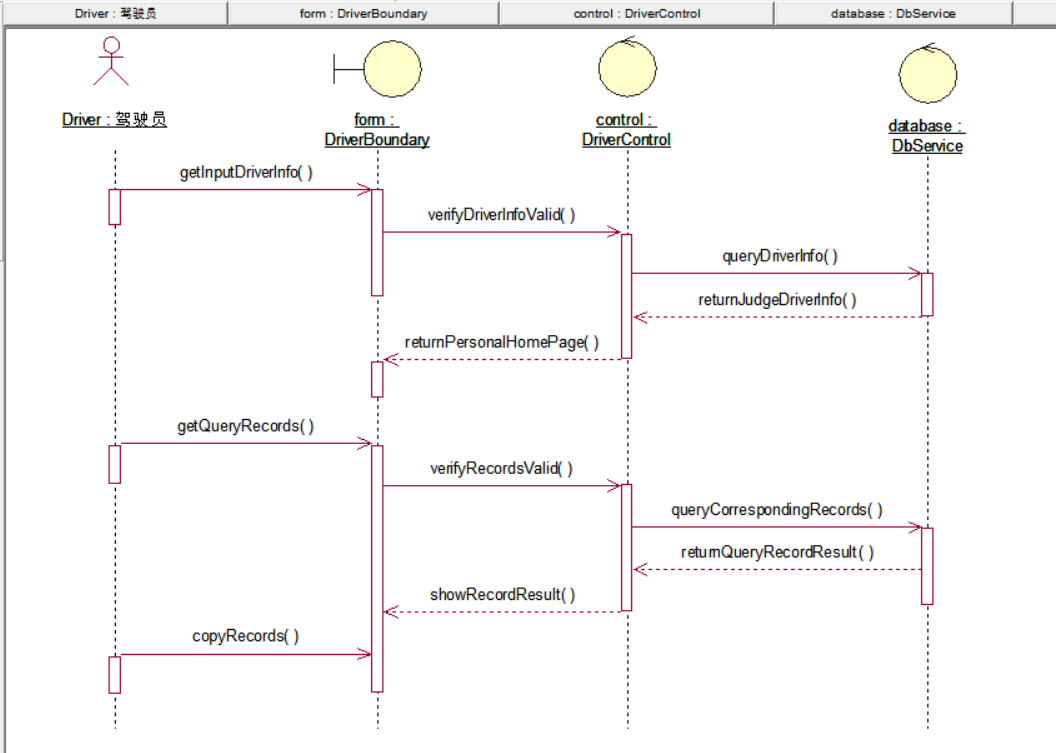


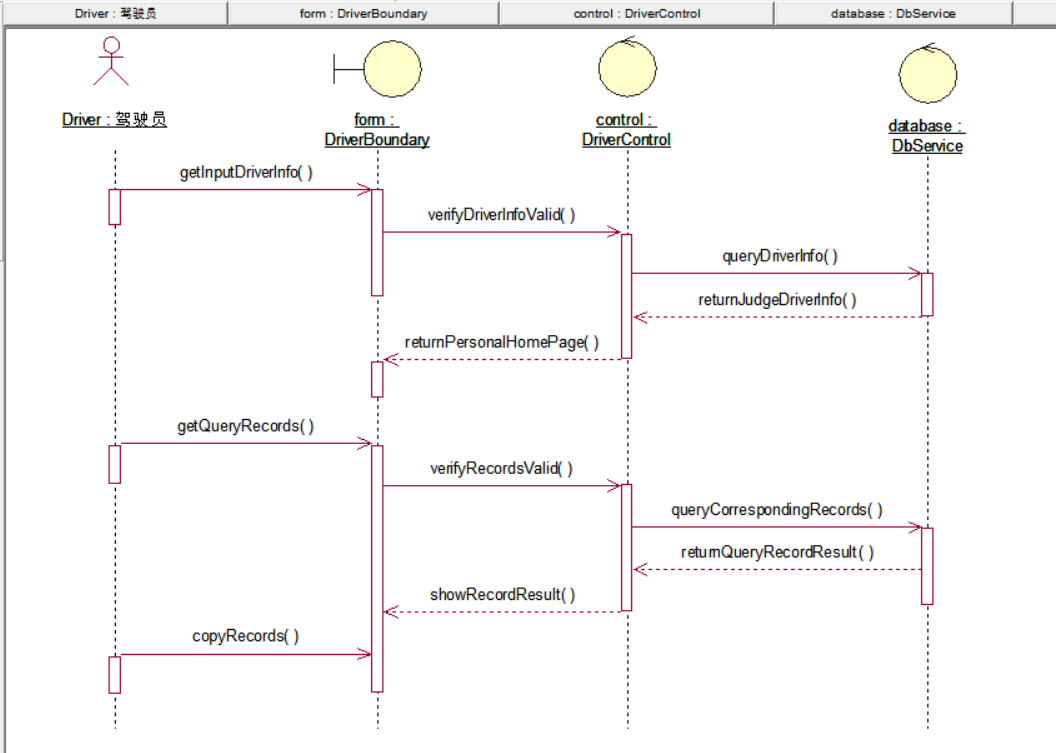
2.一层序列图及协作图——导航



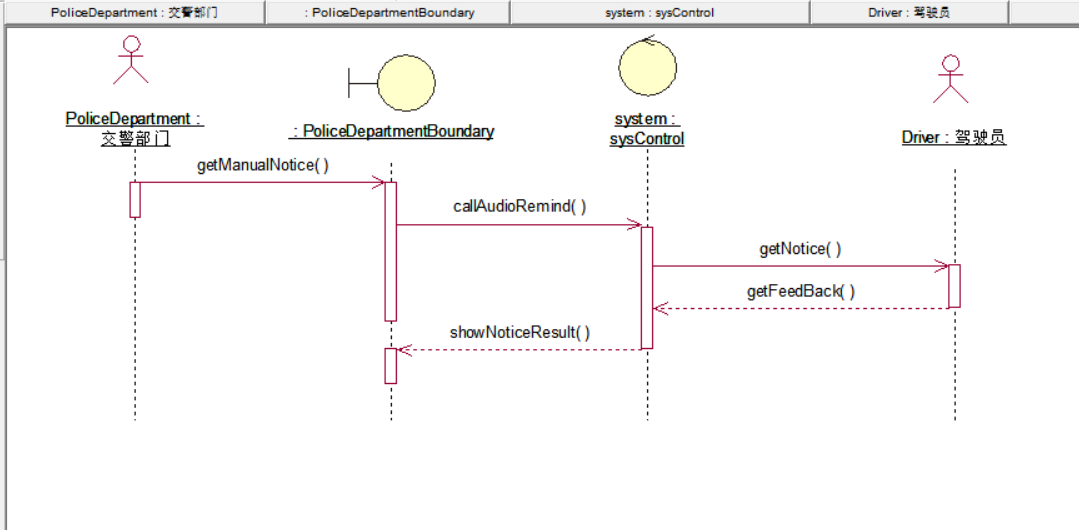


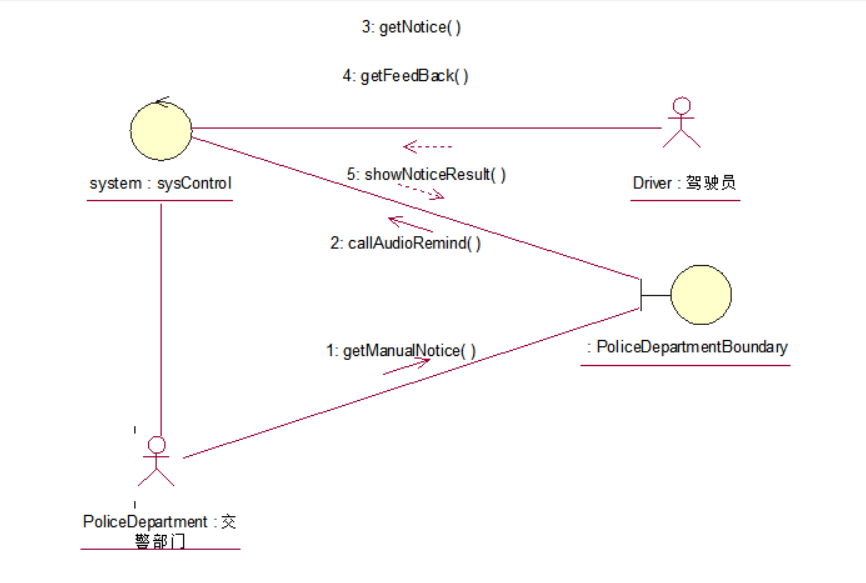
3.一层序列图及协作图——拷贝行车记录



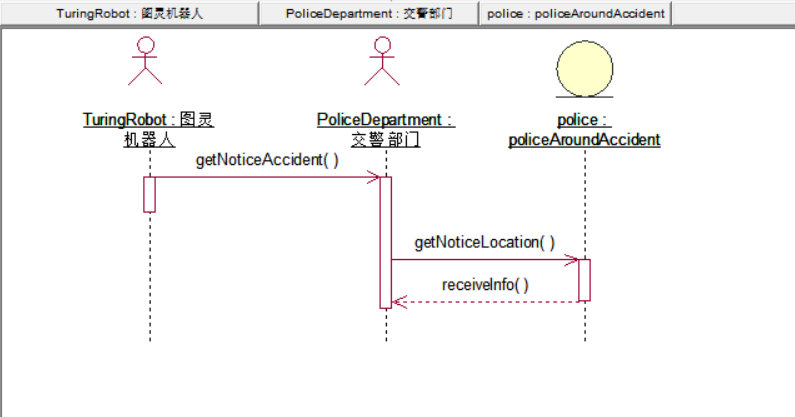


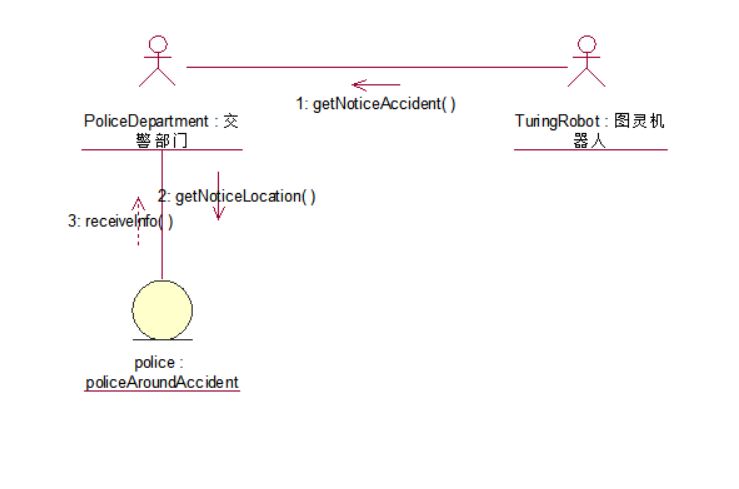
4.一层序例图及协作图——人工语音提醒



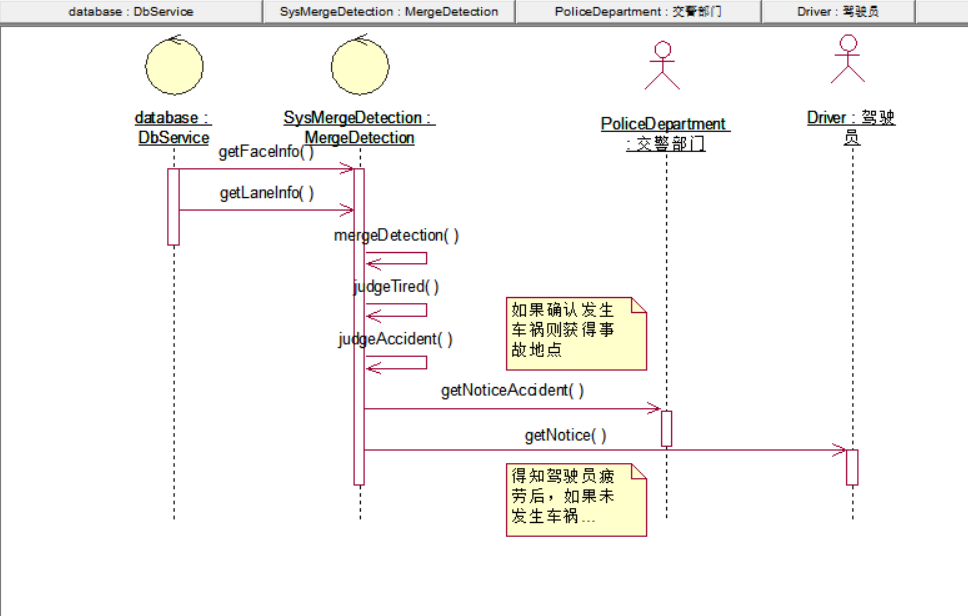


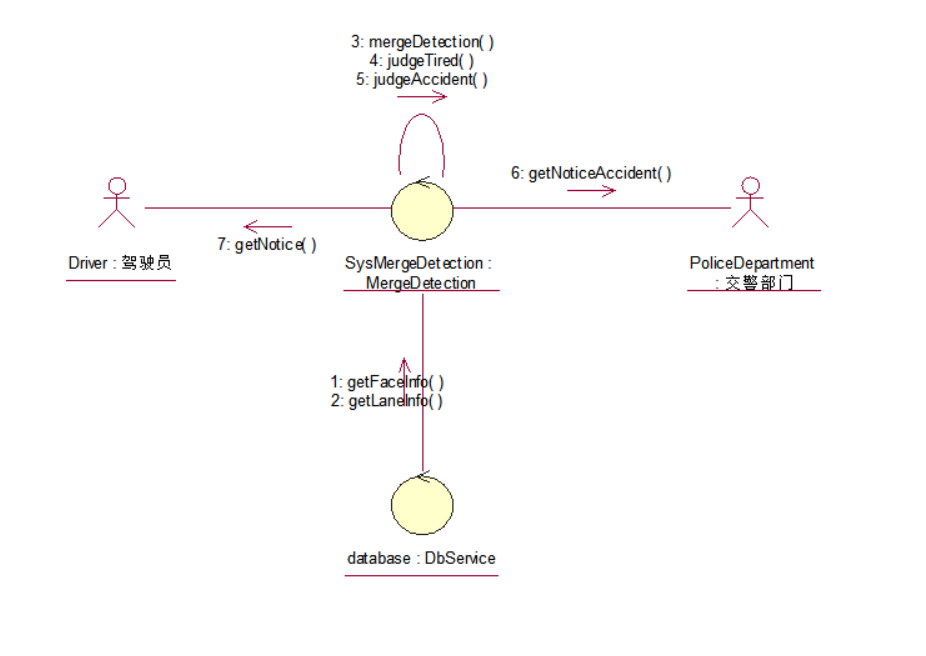
5.一层序列图及协作图——通知事故当地警方



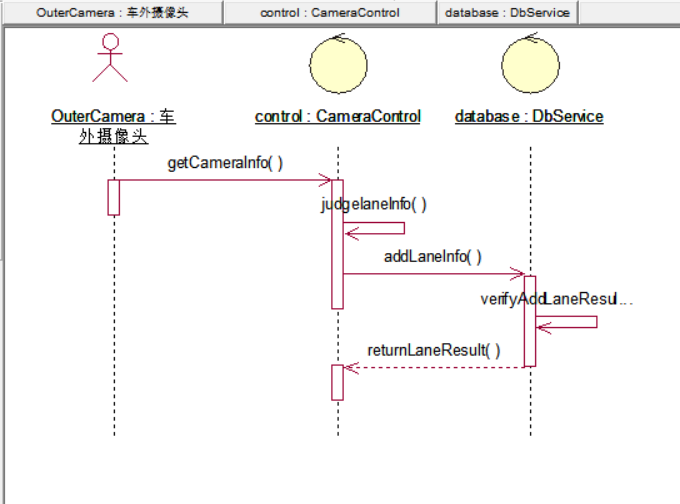


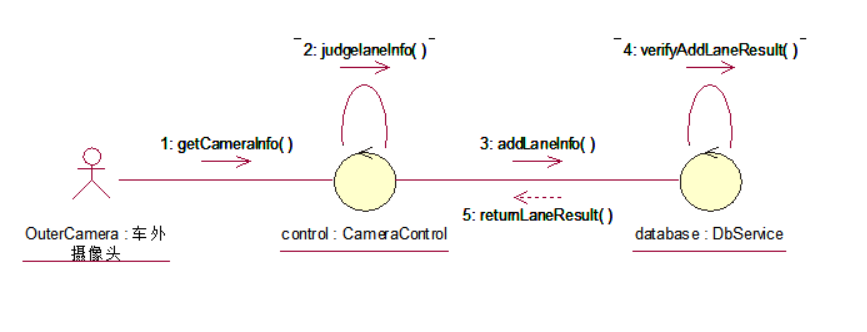
6.一层序列图及协作图——信息融合检测



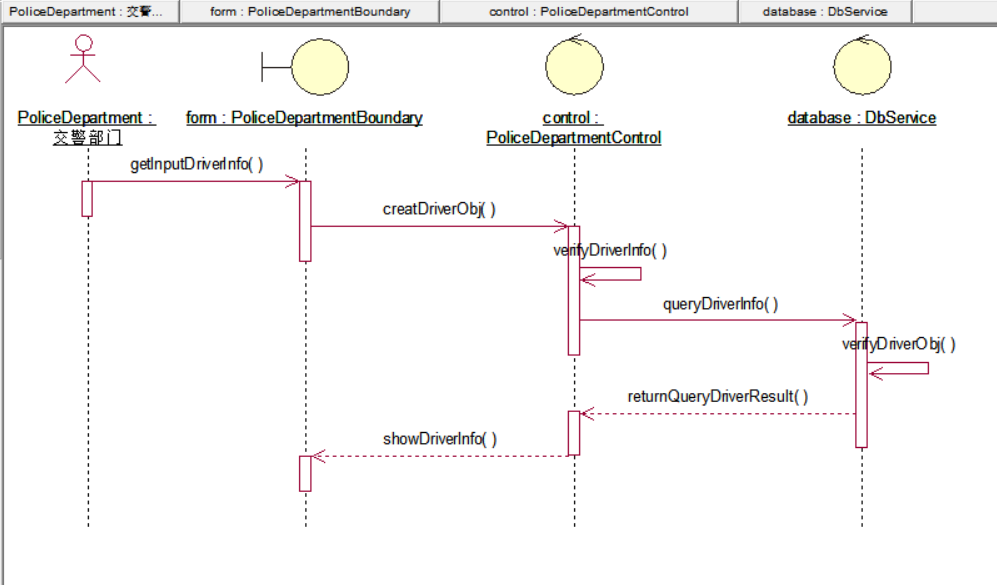


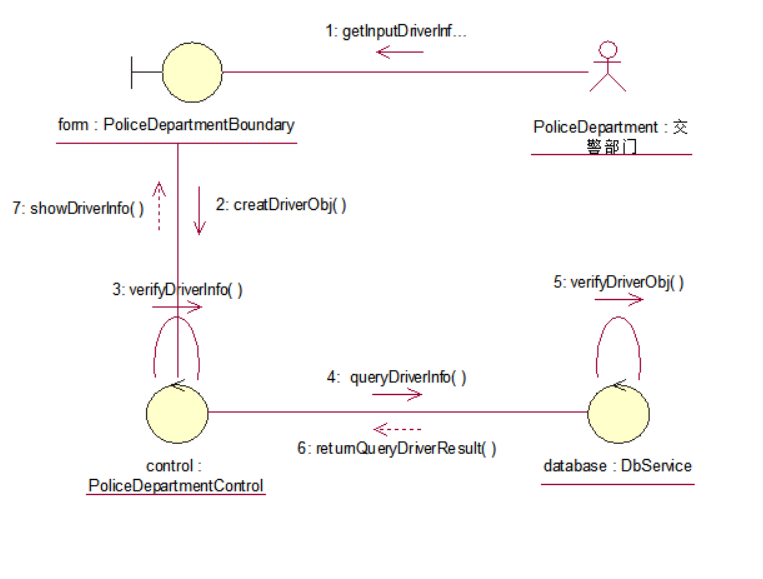
7.二层序列图及协作图——车道信息采集



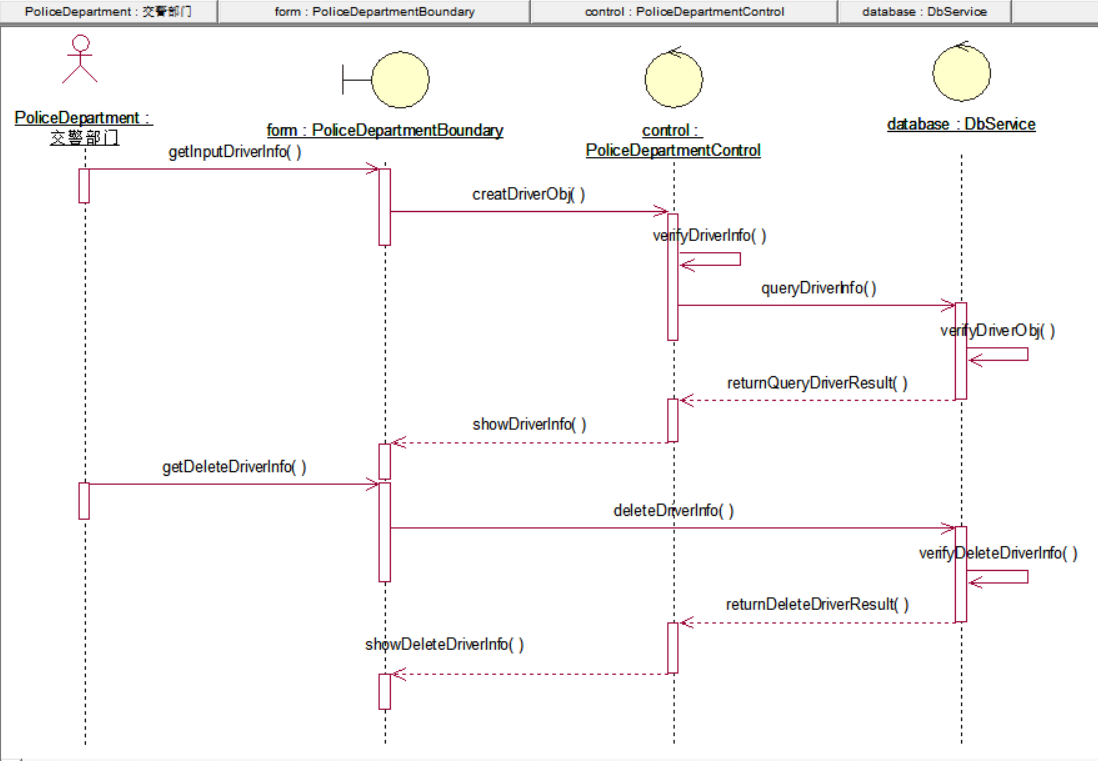


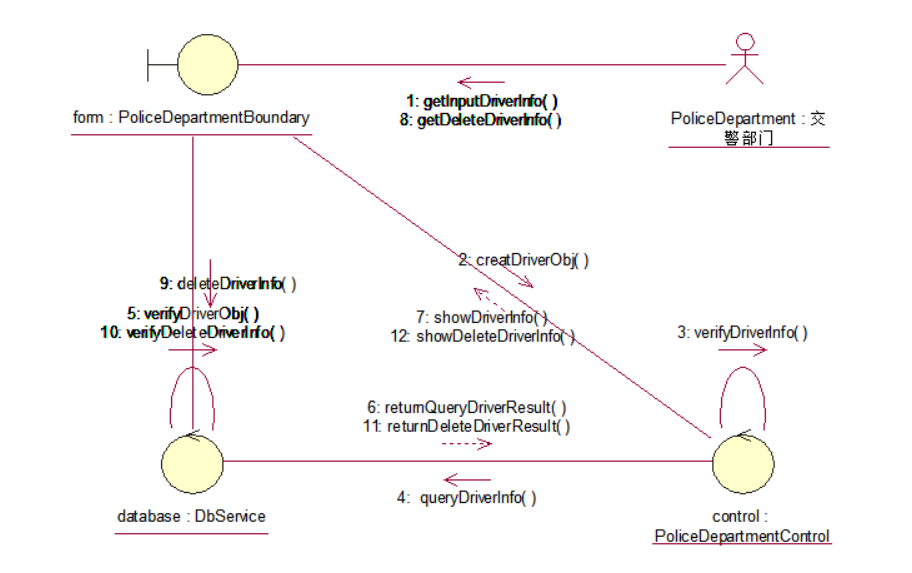
8.二层序列图及协作图——查看驾驶员信息



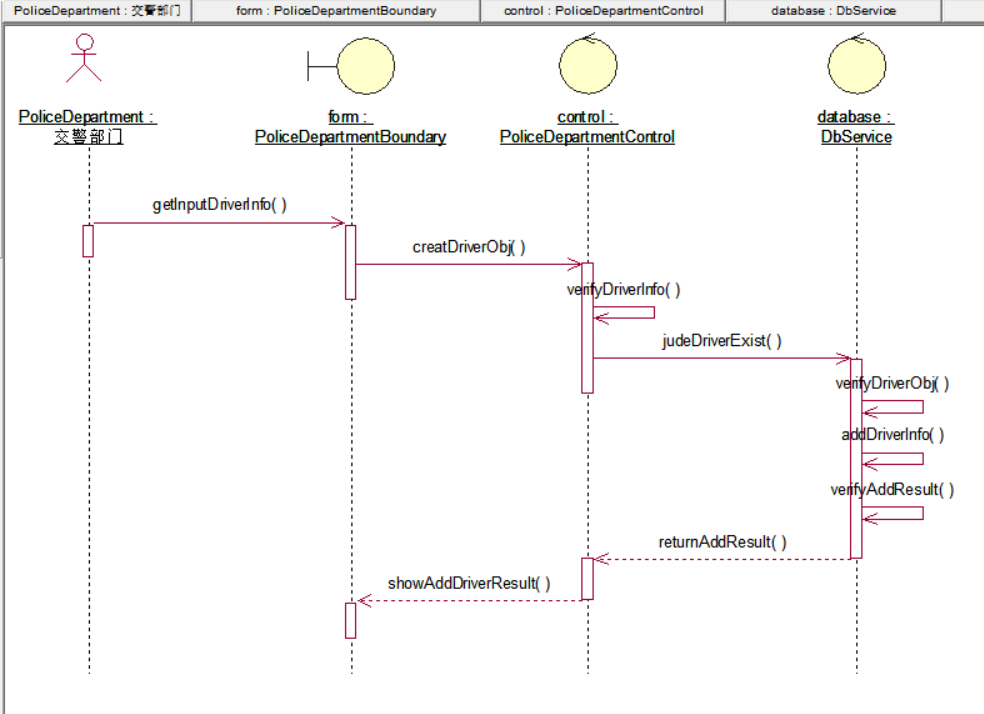


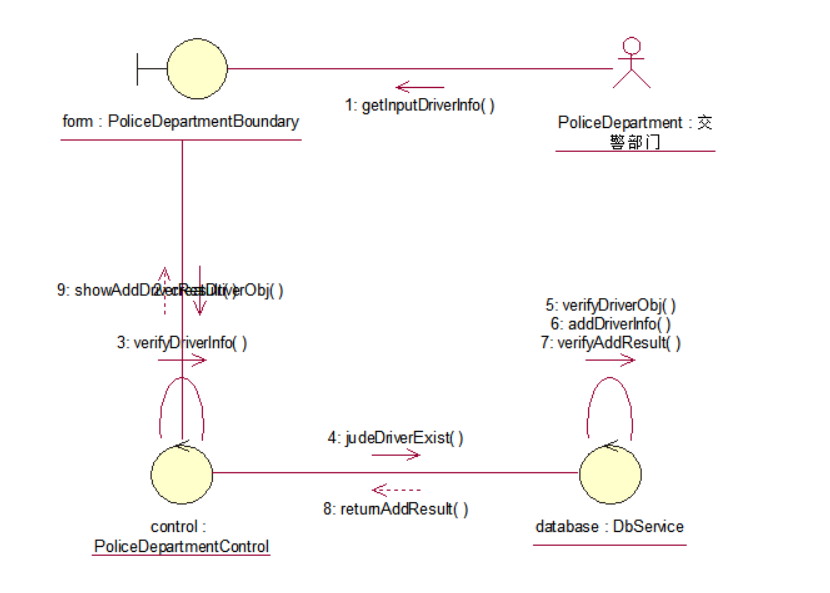
9.二层序列图及协作图——删除驾驶员信息



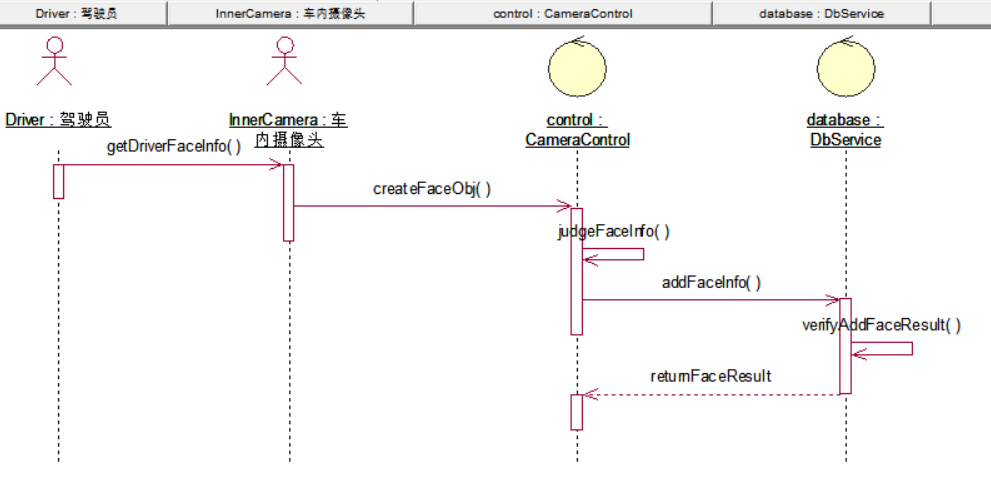


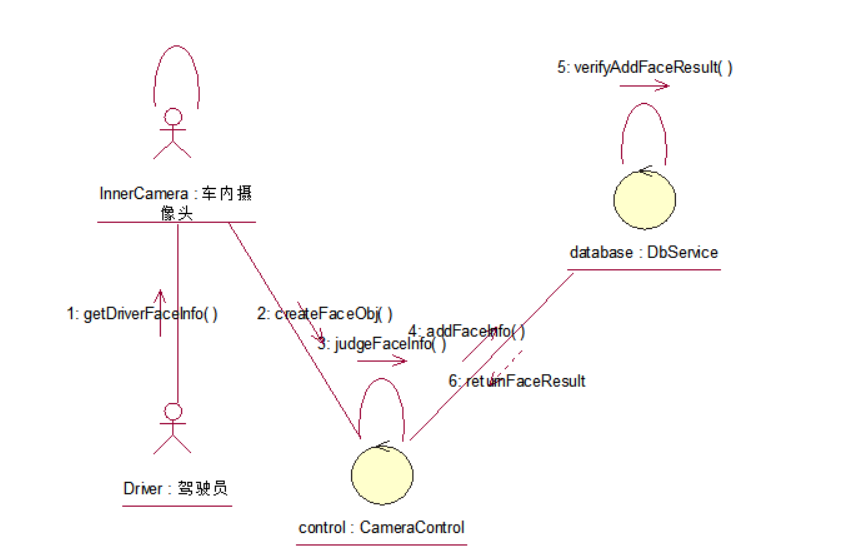
9.二层序列图及协作图——增加驾驶员信息





9.二层序列图及协作图——人脸信息采集





## 状态图

1. 需求分析

针对类，使用状态图建立类对象的生存周期模型来描述对象随时间变化的动态行为。

状态图用于显示状态机（State Mac hine Diagram），重点在与描述状态图的控制流。

状态机描述了门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（Action）*。*

1. 事件流

该部分内容详见《多功能疲劳驾驶检测用例分析文档》。

1. 状态图

1.导航状态图

如图1-1：

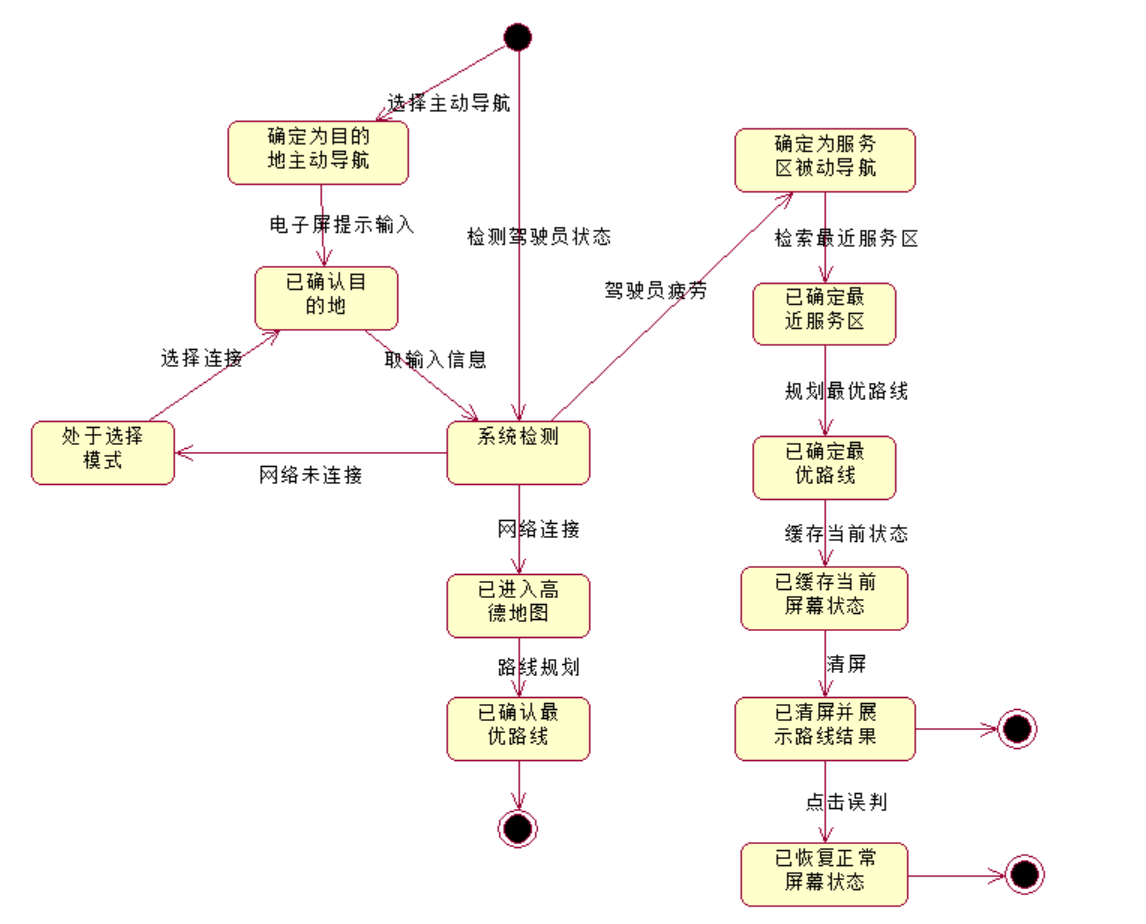


图1-1

登录身份验证状态图

如图1-2：

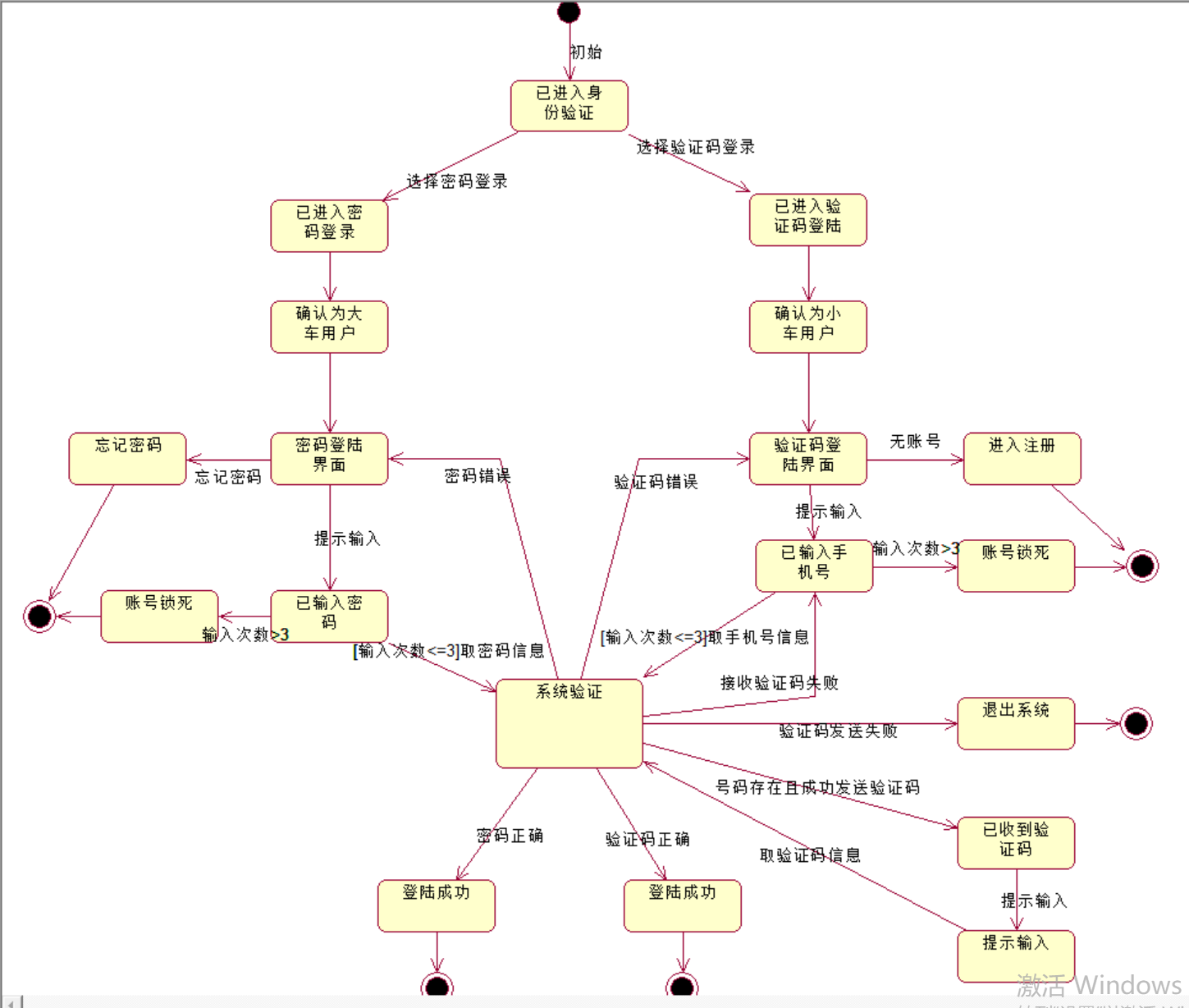


图1-2

对话状态图

如图1-3：

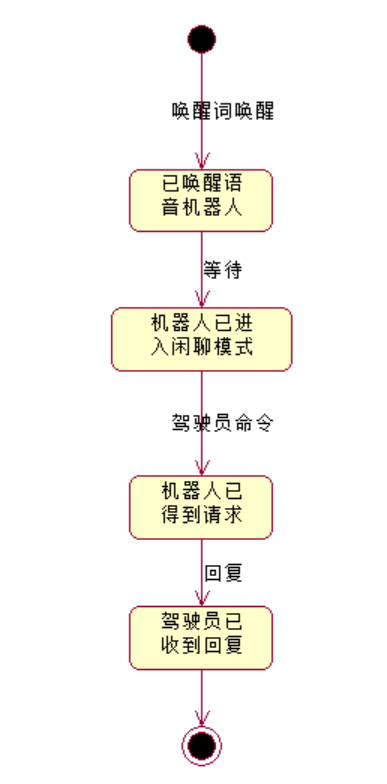


图1-3

行车记录状态图

如图1-4：

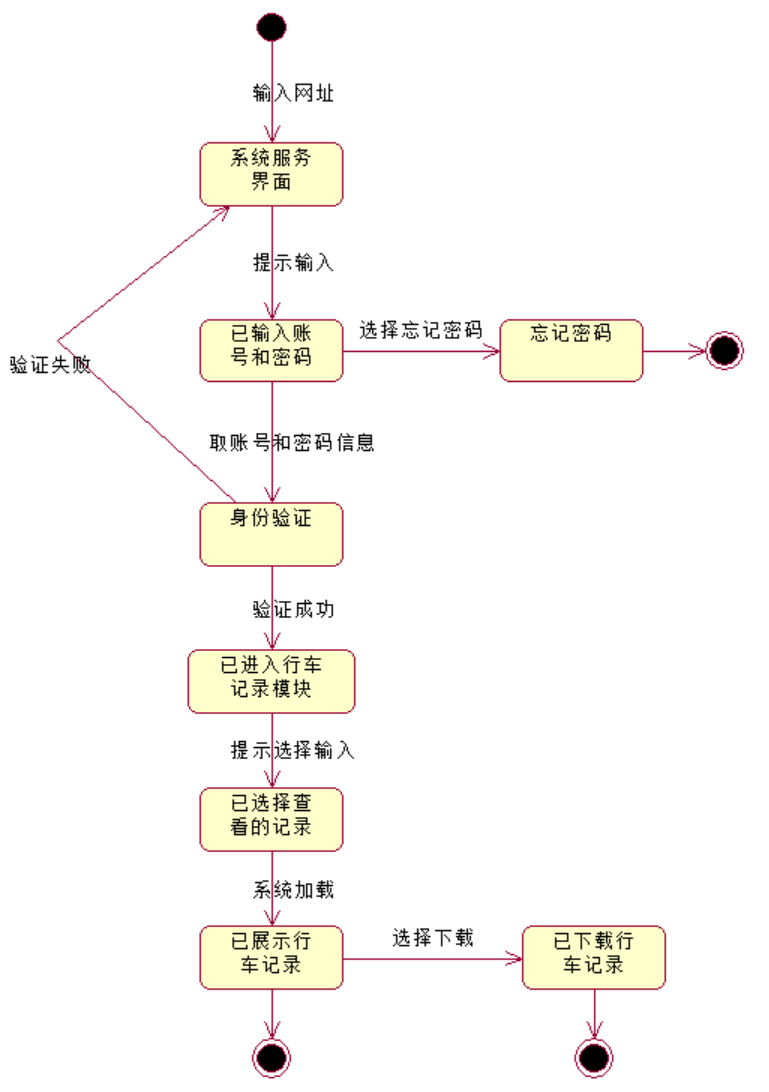


图1-4

激活账号收状态图

如图1-5：

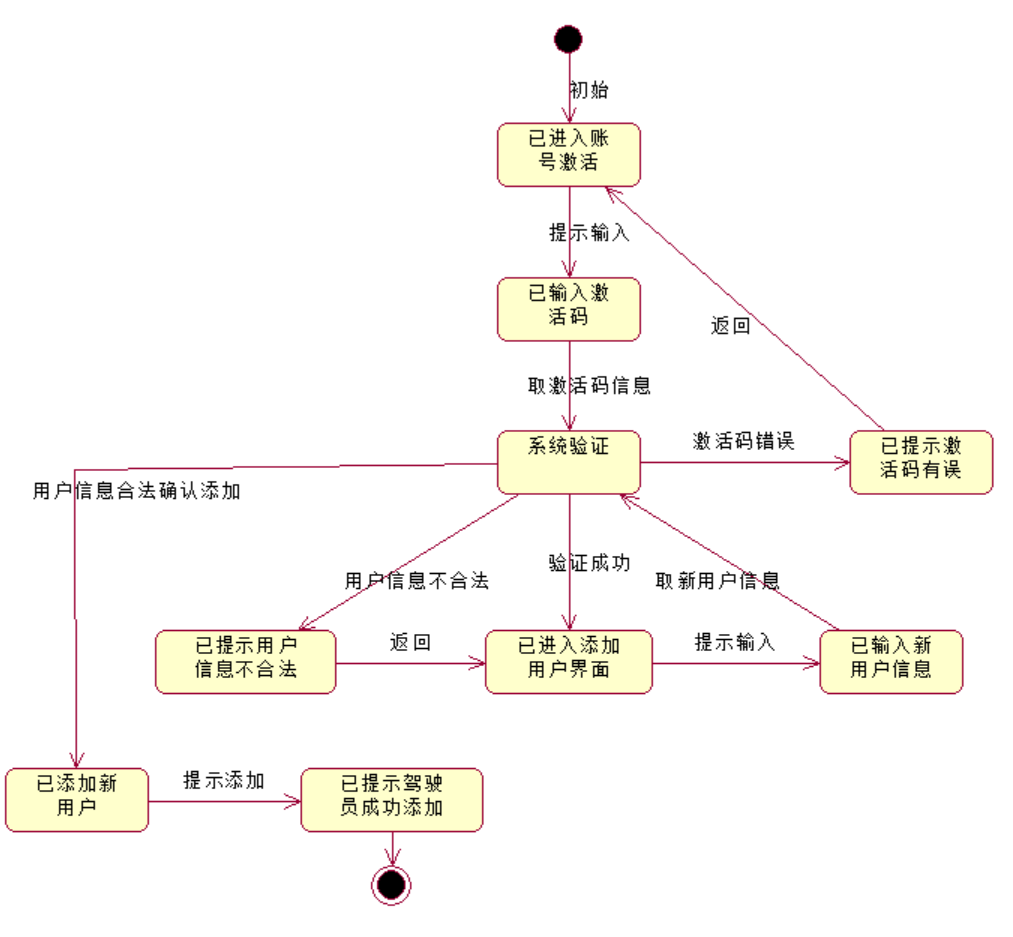


图1-5

通知事故状态图

如图1-6：

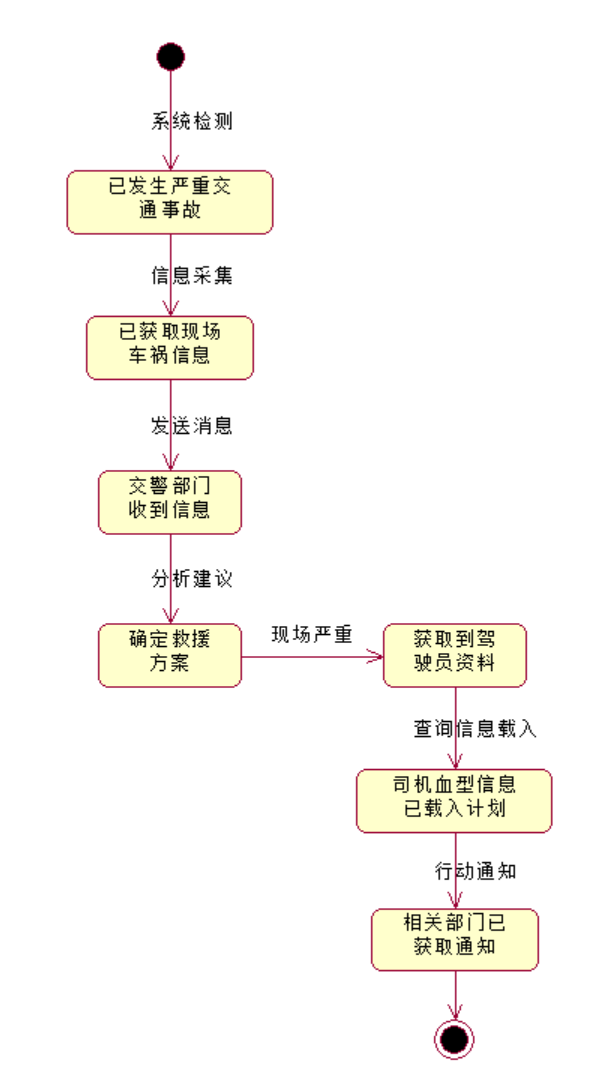


图1-6

信息融合状态图

如图1-7

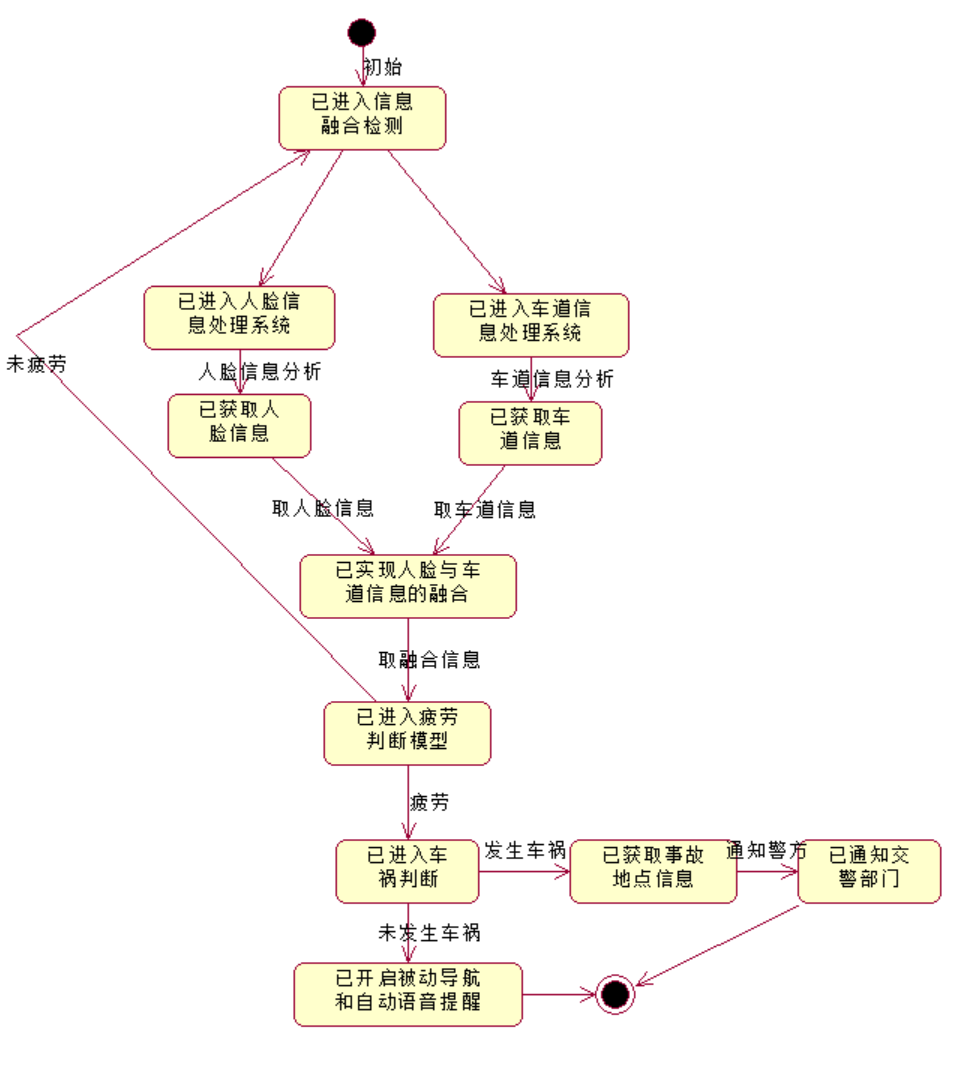


图1-7

语音提醒状态图

如图1-8

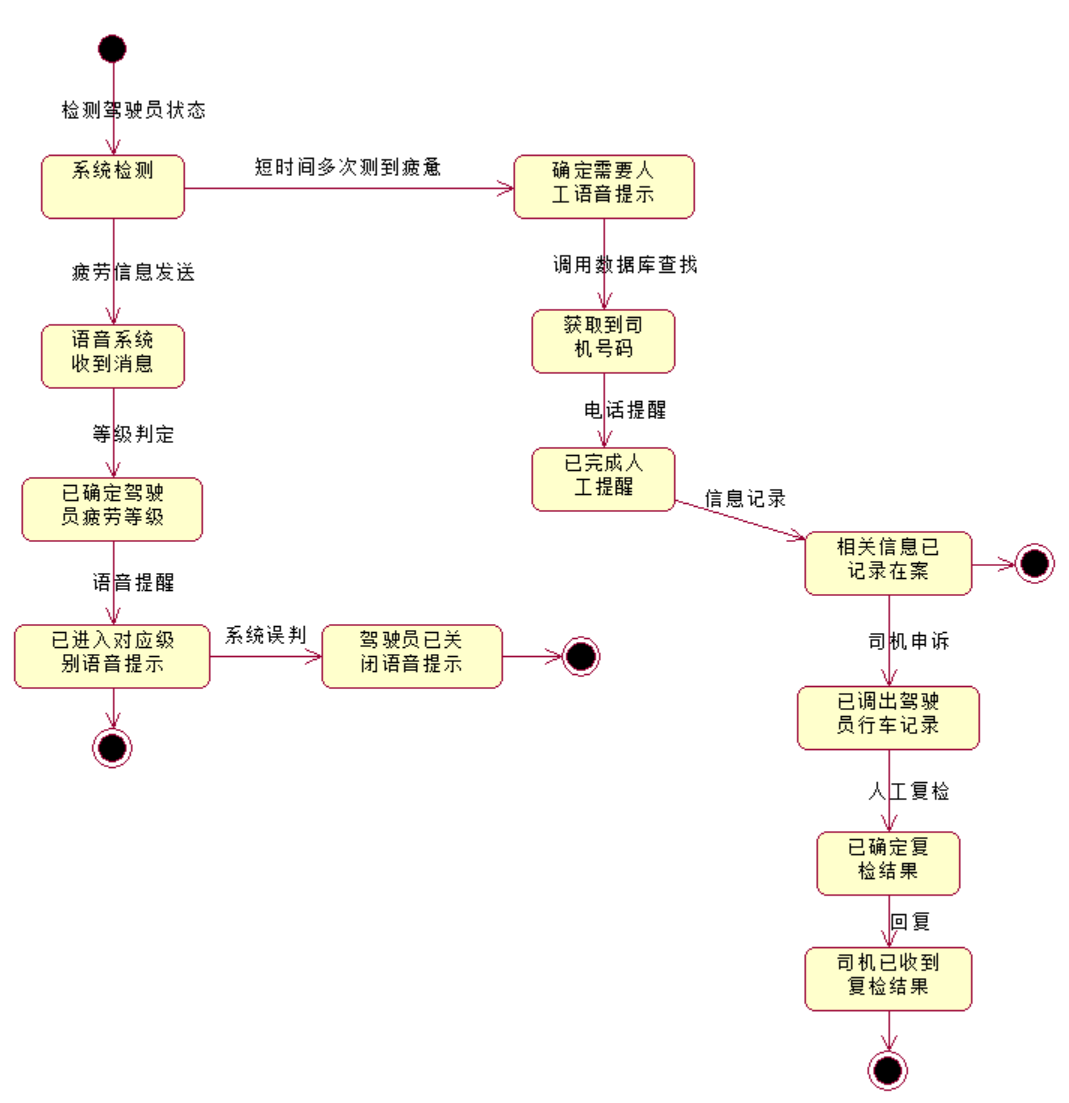


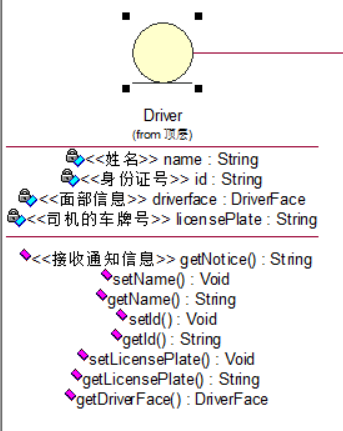
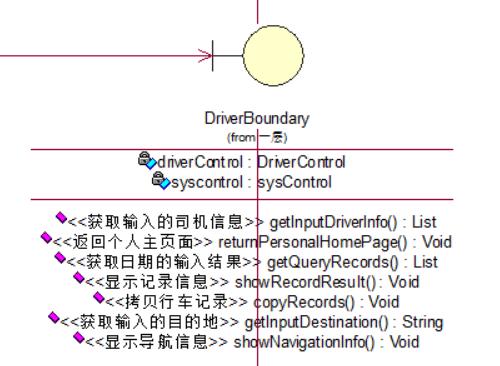
图1-8

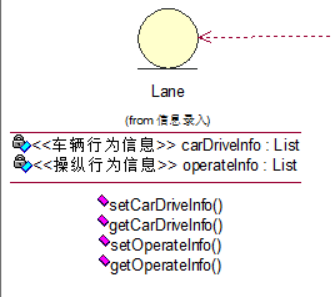
# 逻辑模型设计

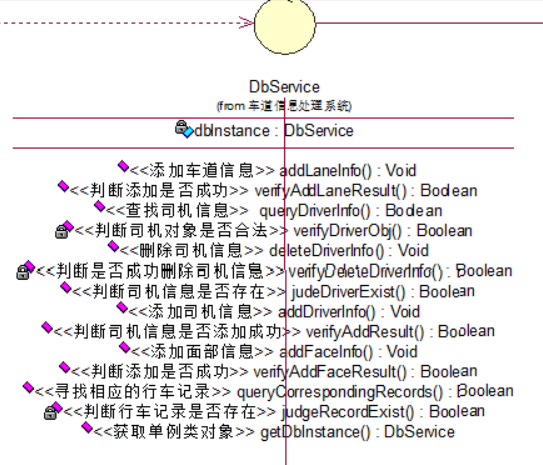
根据用例图和交互图来寻找类，规定类名

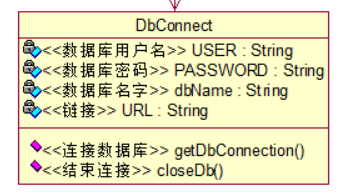
|  |  |  |
| --- | --- | --- |
| 划分 | 类名 | 类说明 |
| Actor | Camera | 抽象类摄像头 |
| FaceCamera | 面部摄像头类 |
| LaneCamera | 车道摄像头类 |
| GouldMap | 高德地图类 |
| TuringRobot | 图灵机器人类 |
| PoliceDepartment | 交警部门类 |
| Controller | CameraControl | 摄像头控制类 |
| DbService | 数据库服务 |
| DriverControl | 驾驶员控制类 |
| MergeDetection | 融合监测类 |
| PoliceDepartmentControl | 交警部门控制类 |
| sysControl | 系统控制类 |
| Entity | Driver | 驾驶员类 |
| DriverFace | 面部类 |
| Lane | 车道类 |
| PoliceAroundAccident | 事故附近交警类 |
| Frame | DriverBoundary | 驾驶员边界类 |
| PoliceDepartmentBoundary | 交警部门边界类 |
| Interface | ISDAO | 数据库操作接口 |
| Service | DbConnect | 数据库连接类 |

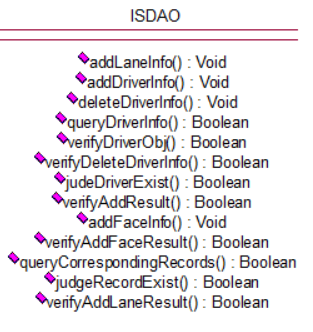
2. 通过交互图和状态图确定各个类的方法属性



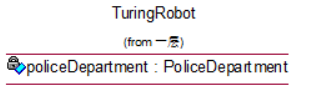
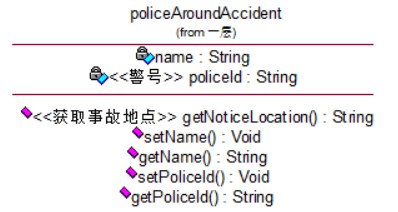


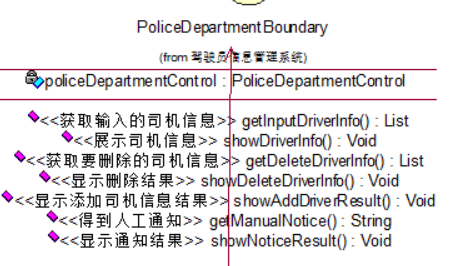
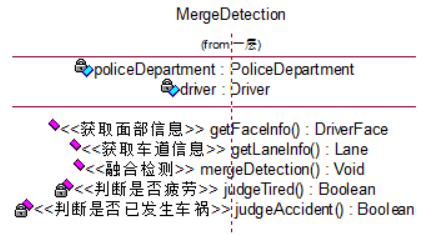


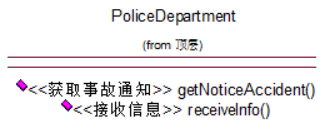


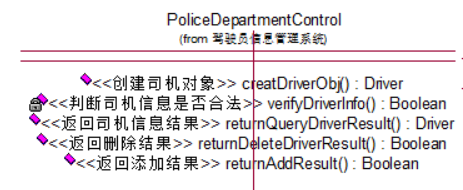




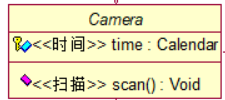






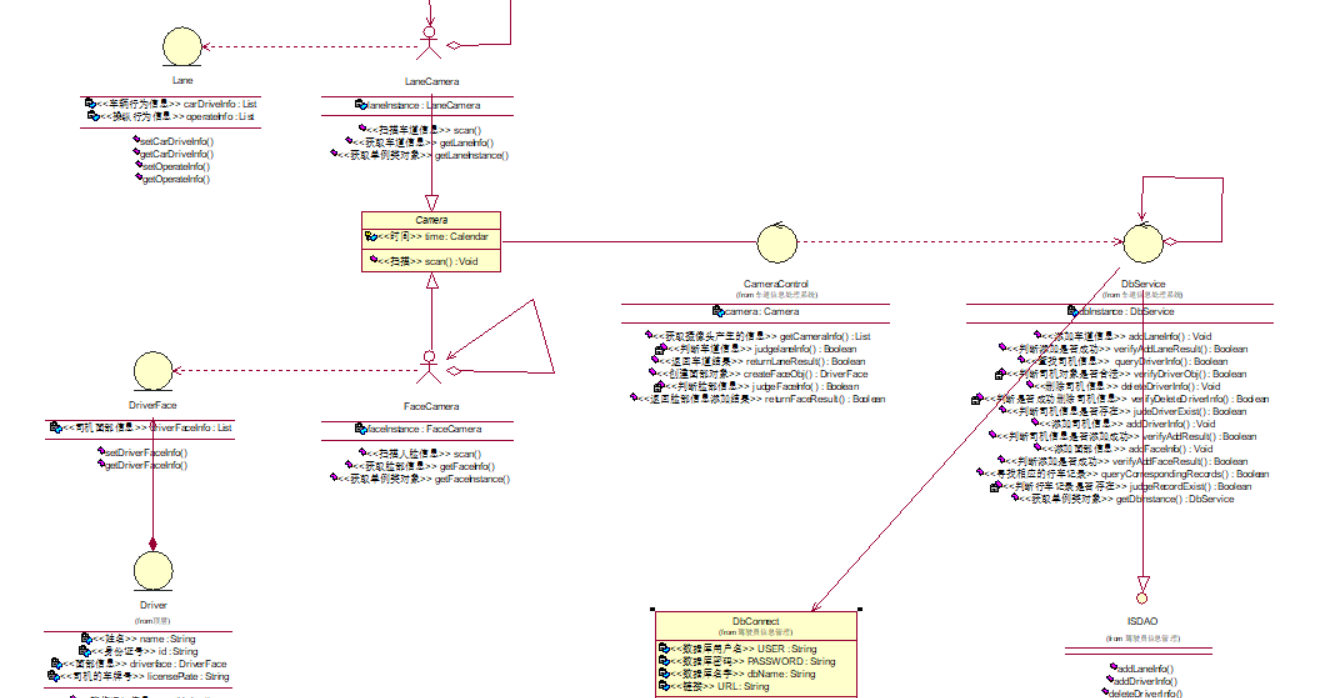




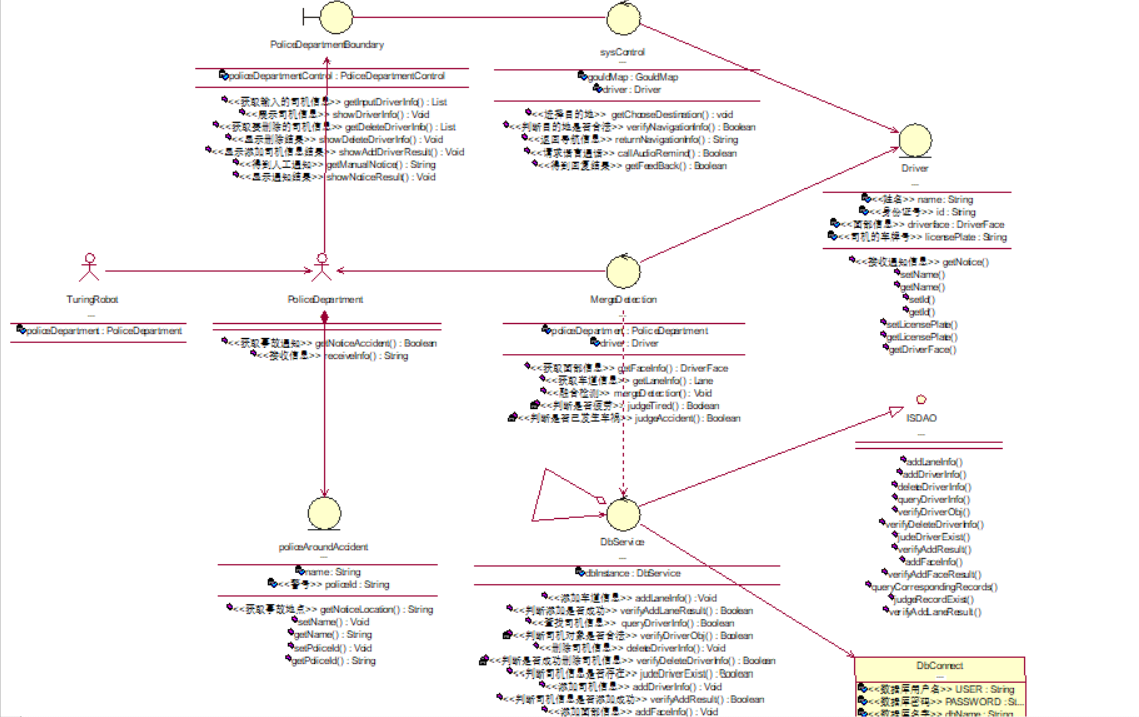


3. 通过交互图和设计模式确定各个类之间的关系，确定设计模式，采用MVC+DAO模式，分包完成UML类图的制作

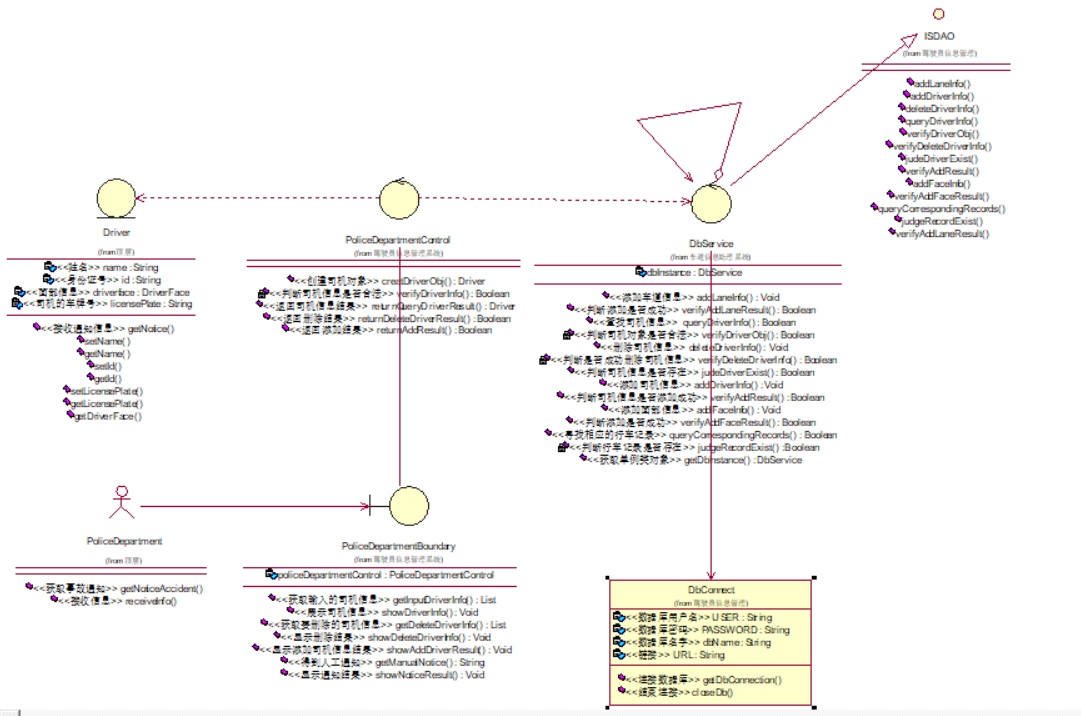
（1）驾驶员信息管理



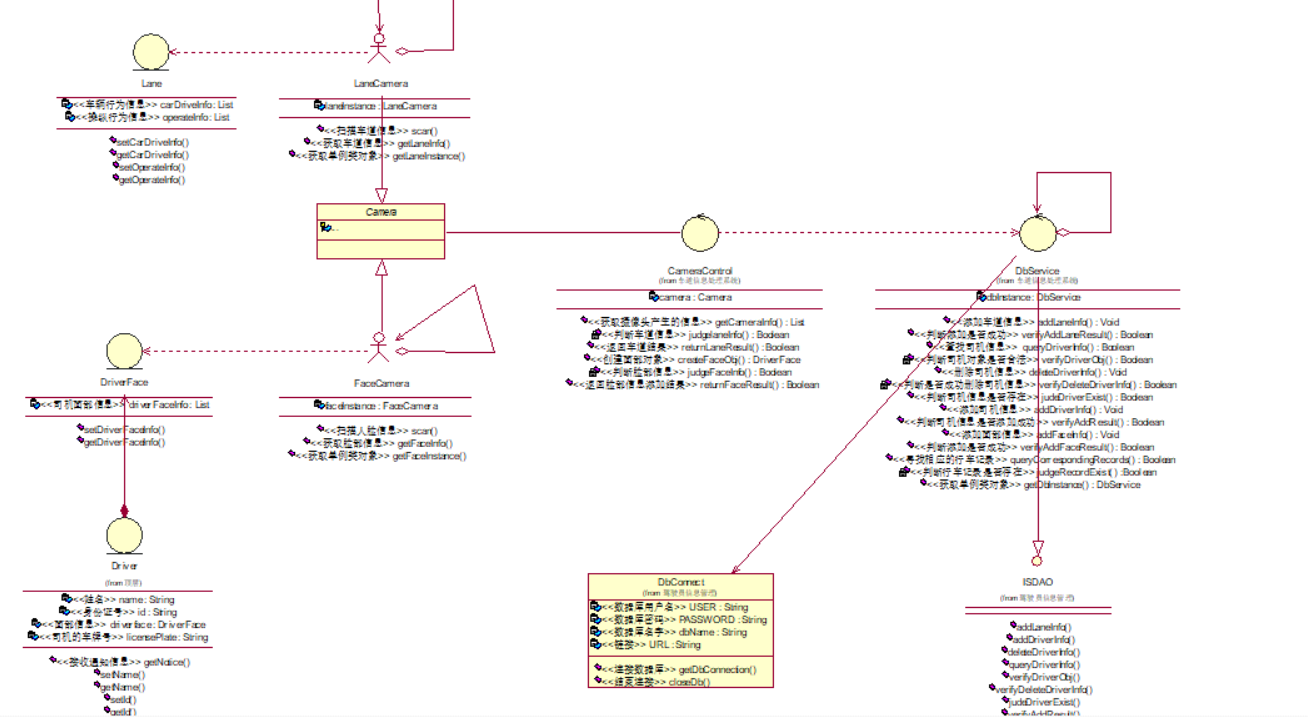
（2）交互提醒



（3）交警部门信息管理



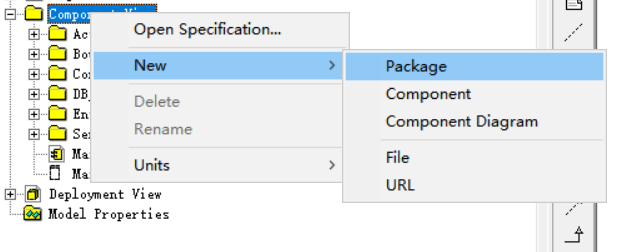
（4）信息录入

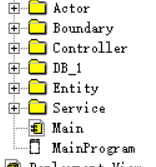


# 物理模型设计

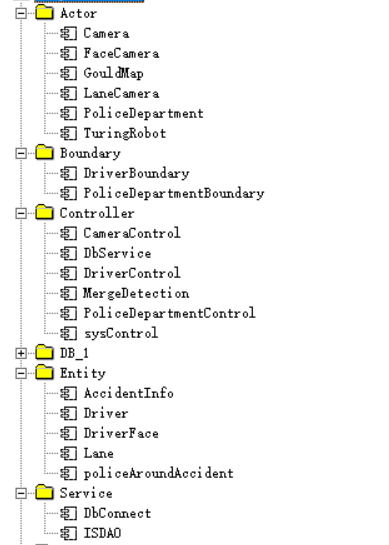
## 组件图

按照下列步骤创建包名，分为Actor（用例），Entity（实体），Boundary（边界），Controller（控制），Service（服务）：

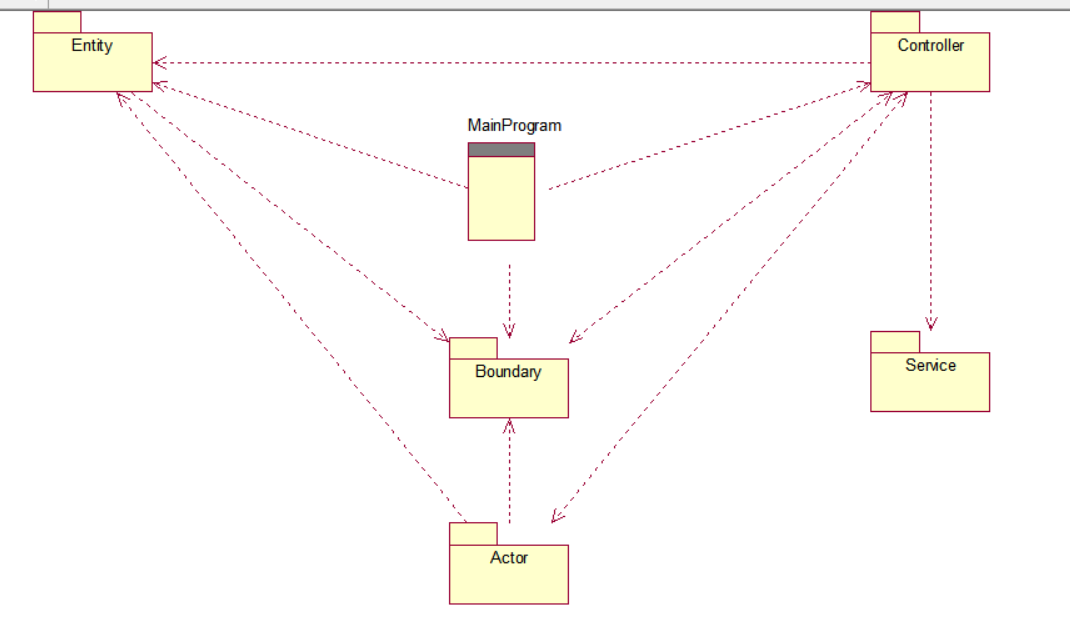




在每个包下面创建与类图对应的组件



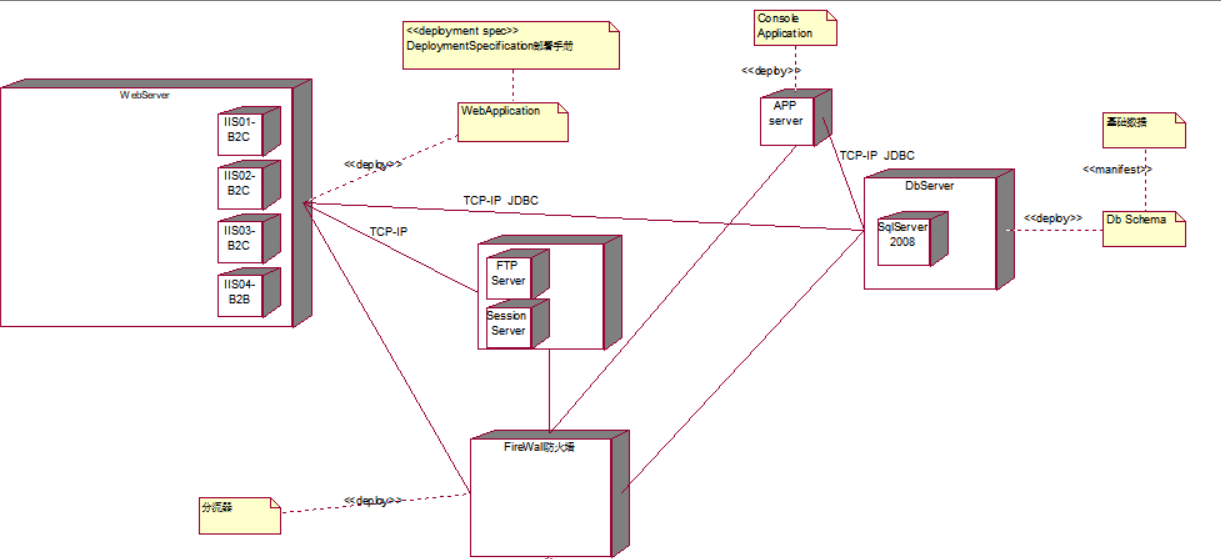
在组件图中，将各个包关联起来



## 部署图

1. 浏览窗口中选择“Deployment View"
2. 在图中添加分别添加Processer(处理器：能够执行软件组件、具有计算能力的节点)和Device(设备（Device）：没有计算能力的节点，通常是通过其接口为外界提供某种服务，例如智能垃圾分类回收系统中的称重器，摄像头和读卡器都是设备)
3. 为节点添加连接关系。

其中数据库服务器和Web服务器之间采用局域网连接通信方式，Web服务器和PC端采用http通信方式



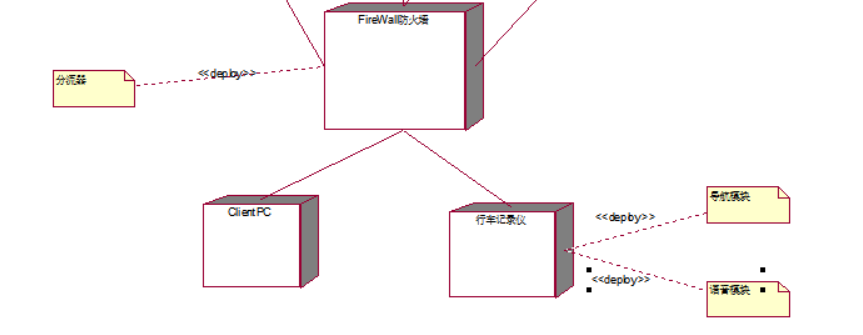
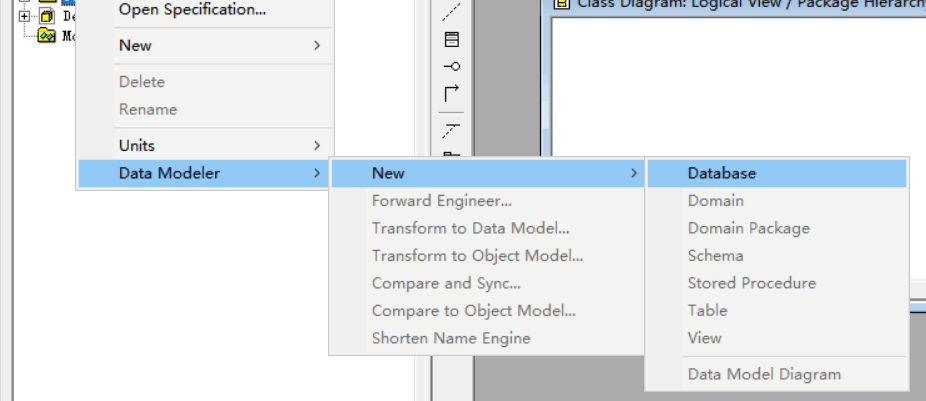


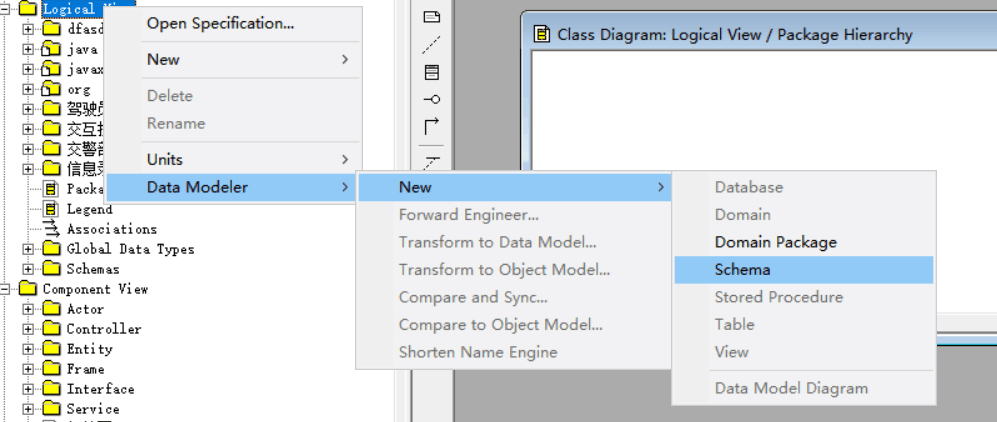
图1 部署图

## 数据模型图

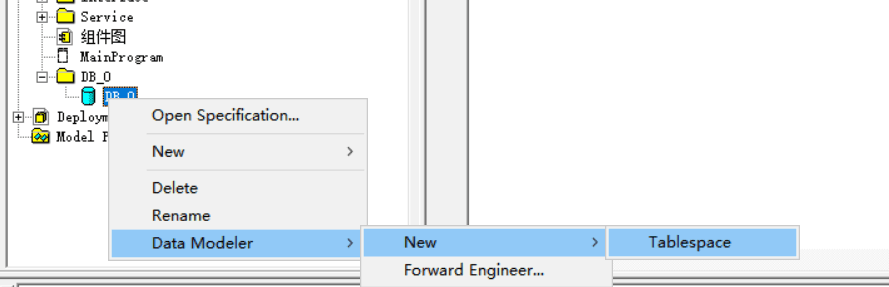
选择“component view”鼠标右键选择Data Modeler--->New--->DataBase,创建数据库



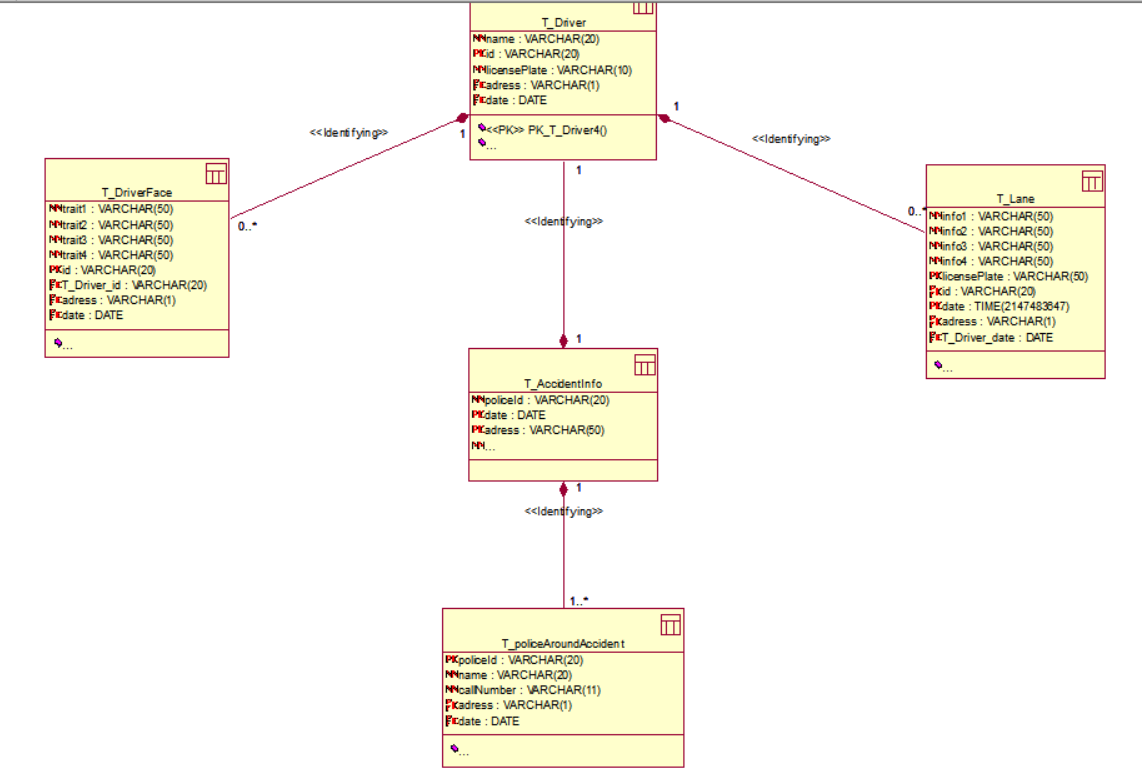
选择“Logic view”鼠标右键选择Data Modeler--->New--->Schema。选择相应的Database



右键数据库，创建表空间



在Schema下创建数据模型图，再创建表与实体对应表，确立表与表之间的对应关系，将表与表关联起来



## 正向工程

（一）检查

在将类转化成代码框架的时候，首先应检查要生成的类有无语法错误。打开要转化的类图，选中要转化的类，单击tools->Java/J2EE->Syntax Check来检查目标代码是否符合规范，比如密码错误。当有错误时，根据log窗口的错误提示校正对应的类。

（二）转化

当要转化的类没有错误时，选中，单击tools -> Java/J2EE -> Generate Code。在进行第一个类的代码文件生成时，需要制定文件的存放路径classpath（点classpath entries下的edit按钮，在出现的界面右上角找new(insert)(删除左边那个),然后在列表里新增的行里选择目标路径），再选择右侧列表里要转换成的包，点assign，即分配给目标classpth，理论上把在一个类图中选择好的类分别分配给多个包。点ok开始生成。如图5-1所示即为导出的文件代码目录结构，图5-2driverInformationManagementr包、图5-3 informationInput包、图5-4 interactionToRemind包、图5-5 policeDepartmentInformationManagement包。以MergeDetection.java为例打开代码文件，内容如图5-6所示。

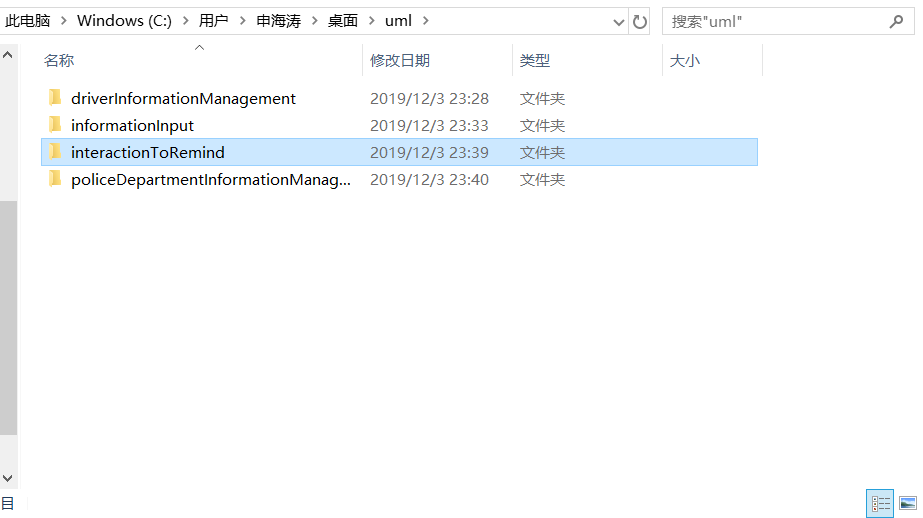


图5-1文件代码结构



图5-2 driverInformationManagementr包

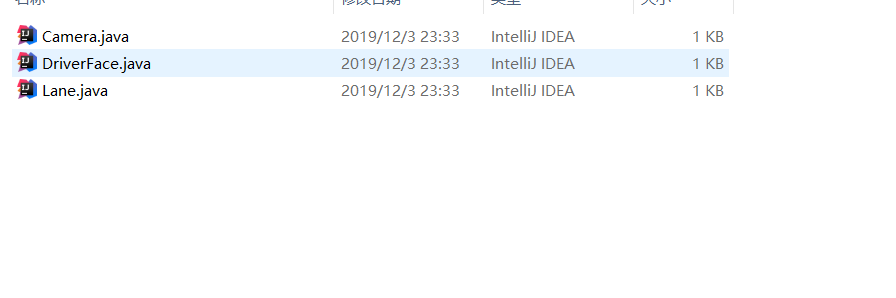


图5-3 informationInput包

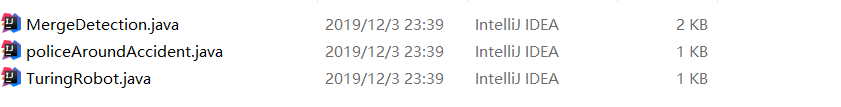




图5-4 interactionToRemind包



图5-5 policeDepartmentInformationManagementr包

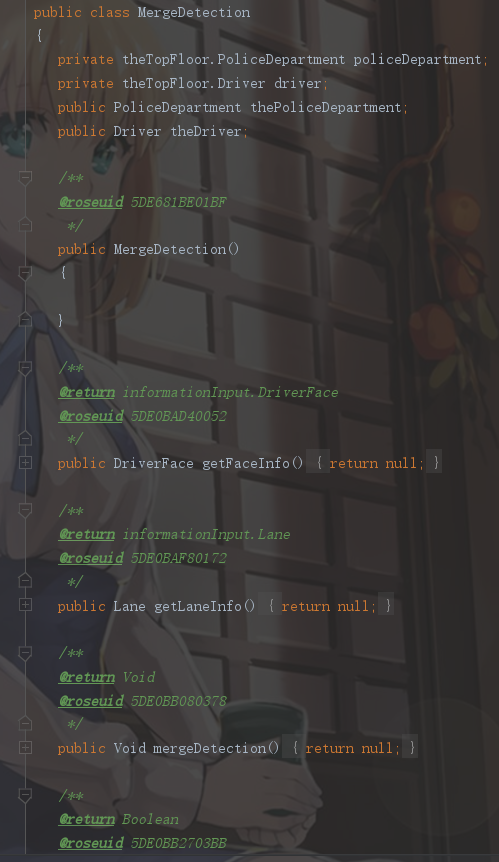


图5 -8

# 总结

用例图是比较重要的，它是直观的反应了系统涉及到的角色、功能以及需求。我第一次做出的用例图只有一层，参与者以及用例杂乱无章的摆放着；并且这么复杂的系统我却不知道如何分层.

自我感觉提取用例是最困难的一步， 主要是我容易把用户的操作步骤误认为是用例，这样得出来的用例图很像流程图，很大程度上不是系统的功能的概述。为此，我一遍一遍的修改用例，先确定好参与者后，我就按照查找到的资料上来发掘用例，比如参与者为什么要使用该系统？参与者是否会在系统中创建、修改、删除、访问、存储数据并且是如何完成这些操作的？参与者是否会将外部的某些事件通知给该系统？系统是否会将内部的某些事件通知该参与者？带着这些问题，该系统的用例逐渐清晰，而且对应的用例规约也逐渐明了。不得不说，找出用例之后，用例图就成功了一大半，对于用例图的分层，按照顶层系统一层主要用例，对于大的用例在继续细分就可以得到他的二级用例图。为了让整个用例建模更加清晰，对于零层一层和二层的用例和用例图，我还单独建了包。

在将用例图逐渐优化的过程中，不仅让我对整个项目更加了解，也对一般的用例建模的流程更加熟练和专业。虽然修改了很多次，但我知道仍然具有不足之处，我将带着这份热情继续学习UML建模。也让我明白了刚开始接触一个新的知识陌生、感觉难以理解是在正常不过的事了，不明白可以问可以查资料但绝不可以放弃，只要静下心来慢慢理解，一步一个脚印的做，一定会有成功的那一刻。

活动图描述了在一个过程中，顺序的/并行的活动及其之间的关系，它是UML用于对系统的动态行为建模的另一种常用工具，可用于描述活动的顺序，展现从一个活动到另一个活动的控制流，常常用于描述业务过程和并行处理过程，活动图在本质上是一种流程图。活动图描述满足用例要求所要进行的活动以及活动间的约束关系，有利于识别并进行活动。

在本系统中，我针对一层用例以及二层用例使用活动图对其进行动态建模。通过多次的修改，完善，对活动图的绘制的基本图形元素和基本要素都有了一定程度的掌握。根据用例的事件流，分析出各个活动的步骤以及联系，再利用各个活动图的图形要素将不同的活动按照实现顺序进行连接，充分直观的体现出各个用例中活动的执行流程。

交互图是对用例图个每个用例的每个操作进行细化，用顺序图具体的划分出来。顺序图强调消息的时间顺序，适合描述实时系统，而交互图则是描述对象之间的关系，同时，通过对Rose软件的学习以及使用，让我对这点的理解更加深刻。这两个图在语义上是等价的，所以我在绘制中也是绘制了其中一种，然后通过软件，让它自动转换成另一种图，这一点很方便，但这一点的不好之处在于，对于这两个图的理解程度就不同了，对自己绘制的顺序图的理解要明显比协作图的理解要深，所以在学习的过程中，我觉得还是两个图都自己画最好了。这样对两种图的理解都一样。

状态图是描述类对象可能经历的所有状态的模型图，描述了对象基于事件反应的动态行为。显示实体根据当时的状态做出具体的动作。

状态图的模型组成元素有状态，转换，时间。而状态又由名称和内部动作所组成。转换则包含原状态，目标状态，触发事件，监护条件，执行动作五个元素。

在本系统中，我针对部分较为复杂的用例使用状态图对其进行动态建模。通过多次的修改，完善，对状态图的绘制方法和技巧有了一定程度的掌握。根据用例的事件流，抓住对象的各个状态，分析出各个状态之间的转换条件，最终利用状态图将用例过程中的各个状态的转变清晰明了的呈现。

组件图：从理解到构思再到画出成品，前前后后花了比较多的时间，也走了不少弯路。在上网查资料的过程中，了解到组件与类之间的异同点，但不知道如何从一个复杂的系统中提炼出组件并精确的表示出他们之间的依赖关系。在思路都是迷茫和错乱的情况下，我静下心来慢慢的理解组件的具体含义以及提炼方法，并参考类的建模时创建的包、类以及接口。参考了不少优秀范例后，我手动画出了系统的组件图。由于系统较为复杂，我还是将系统的组件图分了层，如图6-1组件结构所示是我最早画出来的组件图的结构。图6-2系统组件图、6-3用户操作组件图、6-4管理员操作组件图、6-5管理员管理组件图以及与管理员管理组件图类似的用户管理组件图、垃圾分类信息管理组件图和垃圾清洁与运输管理组件图也是最早画出来的组件图。当然，这些图虽然是建立在类图也就是系统需求的基础上，但并不能准确的结合类图反应出系统的物理实现。后来，在正向工程导出代码文件时，才发现rational rose会自动将逻辑视图（Logical View）中的包以及包中的类或接口转换成组件视图（Component View）中的包以及包中的组件。于是就可以直接利用自动生成的包进行系统组件图的绘制。

虽然在画组件图的过程中走了不少弯路，费了不少无用功以及花了不少的冤枉时间，但我觉得还是值得的。这些经历也让我的实践动手能力有提高了一些，在不理解新知识的前提下，依旧可以沉住气慢慢寻找灵感，借鉴网上的优秀范例解决自己项目中出现的问题。我将继续将这种精益求精的学习劲头应用到学习生活的方方面面，将书本上的理论知识与实际操作中遇到的问题结合起来，不断突破自己，完善自己的作品。

数学模型图：了解到通过Rose绘制数据模型图的方便，只需要根据类图，选择自己需要存入数据库中的实体类，将其设置为Persistent，然后创建数据库组建，然后创建Schema，前面设置了的类图中的实体类就自动创建出来相对应的表，然后创建新的数据模型图，直接将Schema中的表拖入数据模型图，然后再考虑表与表之间的联系，通过箭头，将有联系的表联系在一起。

理解正向工程的含义以及具体操作并不困难，但真正困难的是将智能垃圾分类与回收这个复杂的系统正向工程分析，并导出代码。在听完课堂上老师对于双向工程和导出代码的讲解后，在实际操作中又会遇到老师没有讲过的问题。

虽然解决问题的过程很令人百思不得其解甚至精神有点错乱，但只要认真阅读错误的信息以及积极寻找解决方案，一定会有解决的那一刻。通过这次学习以及团队之间的合作，我愈加体会到集体凝聚力的力量，它催人奋进，给予智慧，提供帮助，我也会更加努力，迎接每一个困难与挑战。

# 参考文献

1. 《面向对象方法 工程实训教材》 李建明 刘晓强 主编
2. 《大象:Thinking in UML》 谭云杰 主编